Subject: [PATCH] kthread: saa7134-tvaudio.c
Posted by Sukadev Bhattiprolu on Tue, 29 Aug 2006 21:15:55 GMT
View Forum Message <> Reply to Message

Replace kernel_thread() with kthread_run() since kernel_thread()
is deprecated in drivers/modules.

Note that this driver, like a few others, allows SIGTERM. Not
sure if that is affected by conversion to kthread. Appreciate
any comments on that.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Containers@lists.osdl.org
Cc: Gerd Knorr <kraxel@bytesex.org>

```
 drivers/media/video/saa7134/saa7134-tvaudio.c |   33 ++++++++++++--------------
 drivers/media/video/saa7134/saa7134.h         |    4 ---
 2 files changed, 17 insertions(+), 20 deletions(-)

Index: lx26-18-rc5/drivers/media/video/saa7134/saa7134.h
===================================================================
--- lx26-18-rc5.orig/drivers/media/video/saa7134/saa7134.h 2006-08-29 14:02:44.000000000
-0700
+++ lx26-18-rc5/drivers/media/video/saa7134/saa7134.h 2006-08-29 14:04:21.000000000 -0700
@@ -311,10 +311,8 @@ struct saa7134_pgtable {

 /* tvaudio thread status */
 struct saa7134_thread {
-	pid_t                      pid;
-	struct completion          exit;
+	struct task_struct *       task;
 	wait_queue_head_t          wq;
-	unsigned int               shutdown;
 	unsigned int               scan1;
 	unsigned int               scan2;
 	unsigned int               mode;
Index: lx26-18-rc5/drivers/media/video/saa7134/saa7134-tvaudio.c
===================================================================
--- lx26-18-rc5.orig/drivers/media/video/saa7134/saa7134-tvaudio.c 2006-08-29
14:02:44.000000000 -0700
+++ lx26-18-rc5/drivers/media/video/saa7134/saa7134-tvaudio.c 2006-08-29 14:06:24.000000000
-0700
@@ -28,6 +28,7 @@
 #include <linux/slab.h>
 #include <linux/delay.h>
 #include <linux/smp_lock.h>
+#include <linux/kthread.h>
 #include <asm/div64.h>

 #include "saa7134-reg.h"
@@ -357,7 +358,7 @@ static int tvaudio_sleep(struct saa7134_
  DECLARE_WAITQUEUE(wait, current);

  add_wait_queue(&dev->thread.wq, &wait);
- if (dev->thread.scan1 == dev->thread.scan2 && !dev->thread.shutdown) {
+ if (dev->thread.scan1 == dev->thread.scan2 && !kthread_should_stop()) {
   if (timeout < 0) {
    set_current_state(TASK_INTERRUPTIBLE);
    schedule();
@@ -525,7 +526,7 @@ static int tvaudio_thread(void *data)
  allow_signal(SIGTERM);
  for (;;) {
   tvaudio_sleep(dev,-1);
-  if (dev->thread.shutdown || signal_pending(current))
+  if (kthread_should_stop() || signal_pending(current))
    goto done;

  restart:
@@ -633,7 +634,7 @@ static int tvaudio_thread(void *data)
   for (;;) {
    if (tvaudio_sleep(dev,5000))
     goto restart;
-   if (dev->thread.shutdown || signal_pending(current))
+   if (kthread_should_stop() || signal_pending(current))
     break;
    if (UNSET == dev->thread.mode) {
     rx = tvaudio_getstereo(dev,&tvaudio[i]);
@@ -649,7 +650,6 @@ static int tvaudio_thread(void *data)
  }

  done:
- complete_and_exit(&dev->thread.exit, 0);
  return 0;
 }

@@ -798,7 +798,6 @@ static int tvaudio_thread_ddep(void *dat
  struct saa7134_dev *dev = data;
  u32 value, norms, clock;

- daemonize("%s", dev->name);
  allow_signal(SIGTERM);

  clock = saa7134_boards[dev->board].audio_clock;
```

```
@@ -812,7 +811,7 @@ static int tvaudio_thread_ddep(void *dat

  for (;;) {
   tvaudio_sleep(dev,-1);
-  if (dev->thread.shutdown || signal_pending(current))
+  if (kthread_should_stop() || signal_pending(current))
    goto done;

  restart:
@@ -894,7 +893,6 @@ static int tvaudio_thread_ddep(void *dat
  }

  done:
- complete_and_exit(&dev->thread.exit, 0);
  return 0;
 }

@@ -1004,15 +1002,16 @@ int saa7134_tvaudio_init2(struct saa7134
   break;
  }

- dev->thread.pid = -1;
+ dev->thread.task = NULL;
  if (my_thread) {
   /* start tvaudio thread */
   init_waitqueue_head(&dev->thread.wq);
-  init_completion(&dev->thread.exit);
-  dev->thread.pid = kernel_thread(my_thread,dev,0);
-  if (dev->thread.pid < 0)
+  dev->thread.task = kthread_run(my_thread,dev,dev->name);
+  if (IS_ERR(dev->thread.task)) {
    printk(KERN_WARNING "%s: kernel_thread() failed\n",
-          dev->name);
+                                dev->name);
+   dev->thread.task = NULL;
+  }
   saa7134_tvaudio_do_scan(dev);
  }

@@ -1023,10 +1022,10 @@ int saa7134_tvaudio_init2(struct saa7134
 int saa7134_tvaudio_fini(struct saa7134_dev *dev)
 {
  /* shutdown tvaudio thread */
- if (dev->thread.pid >= 0) {
-  dev->thread.shutdown = 1;
-  wake_up_interruptible(&dev->thread.wq);
-  wait_for_completion(&dev->thread.exit);
+ if (dev->thread.task) {
```

```
+  /* kthread_stop() wakes up the thread */
+  kthread_stop(dev->thread.task);
+  dev->thread.task = NULL;
  }
  saa_andorb(SAA7134_ANALOG_IO_SELECT, 0x07, 0x00); /* LINE1 */
  return 0;
@@ -1034,7 +1033,7 @@ int saa7134_tvaudio_fini(struct saa7134_

 int saa7134_tvaudio_do_scan(struct saa7134_dev *dev)
 {
- if (dev->thread.pid >= 0) {
+ if (dev->thread.task) {
   dev->thread.mode = UNSET;
   dev->thread.scan2++;
   wake_up_interruptible(&dev->thread.wq);
```

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

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
Posted by Dave Hansen on Tue, 29 Aug 2006 21:22:26 GMT
View Forum Message <> Reply to Message

On Tue, 2006-08-29 at 14:15 -0700, Sukadev Bhattiprolu wrote:

```
> @@ -1004,15 +1002,16 @@ int saa7134_tvaudio_init2(struct saa7134
>    break;
>   }
> 
> - dev->thread.pid = -1;
> + dev->thread.task = NULL;
>   if (my_thread) {
...
```

This is _really_ minor, but I think dev is kzmalloc()'d.  I haven't examined it closely enough to tell if these devices get reused, but this one _might_ be unnecessary.  Certainly no big deal either way, and it certainly doesn't make anything worse.

-- Dave

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

Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
Posted by Andrew Morton on Tue, 29 Aug 2006 21:39:02 GMT
View Forum Message <> Reply to Message

On Tue, 29 Aug 2006 14:15:55 -0700
Sukadev Bhattiprolu <sukadev@us.ibm.com> wrote:

>
> Replace kernel_thread() with kthread_run() since kernel_thread()
> is deprecated in drivers/modules.
>
> Note that this driver, like a few others, allows SIGTERM. Not
> sure if that is affected by conversion to kthread. Appreciate
> any comments on that.
>

hm, I think this driver needs more help.

- It shouldn't be using signals at all, really. Signals are for
  userspace IPC. The kernel internally has better/richer/faster/tighter
  ways of inter-thread communication.

- saa7134_tvaudio_fini()-versus-tvaudio_sleep() looks racy:

```
 if (dev->thread.scan1 == dev->thread.scan2 && !dev->thread.shutdown) {
  if (timeout < 0) {
   set_current_state(TASK_INTERRUPTIBLE);
   schedule();
```

  If the wakeup happens after the test of dev->thread.shutdown, that sleep will
  be permanent.

So in general, yes, the driver should be converted to the kthread API -
this is a requirement for virtualisation, but I forget why, and that's the
"standard" way of doing it.

- The signal stuff should go away if at all possible.

- the thread.shutdown field should go away and be replaced by
  kthread_should_stop().

- the tvaudio_sleep() race might need some attention (simply moving the
  set_current_state() to before the add_wait_queue() will suffice).

- the complete_and_exit() stuff might (should) no longer be needed -
  kthread_stop() does that.

Sorry ;)

```
>  2 files changed, 17 insertions(+), 20 deletions(-)
>
> Index: lx26-18-rc5/drivers/media/video/saa7134/saa7134.h
> ===================================================================
> --- lx26-18-rc5.orig/drivers/media/video/saa7134/saa7134.h 2006-08-29 14:02:44.000000000
-0700
> +++ lx26-18-rc5/drivers/media/video/saa7134/saa7134.h 2006-08-29 14:04:21.000000000
-0700
> @@ -311,10 +311,8 @@ struct saa7134_pgtable {
>
>  /* tvaudio thread status */
>  struct saa7134_thread {
> - pid_t                      pid;
> - struct completion          exit;
> + struct task_struct *       task;
>   wait_queue_head_t          wq;
> - unsigned int               shutdown;
>   unsigned int               scan1;
>   unsigned int               scan2;
>   unsigned int               mode;
> Index: lx26-18-rc5/drivers/media/video/saa7134/saa7134-tvaudio.c
> ===================================================================
> --- lx26-18-rc5.orig/drivers/media/video/saa7134/saa7134-tvaudio.c 2006-08-29
14:02:44.000000000 -0700
> +++ lx26-18-rc5/drivers/media/video/saa7134/saa7134-tvaudio.c 2006-08-29
14:06:24.000000000 -0700
> @@ -28,6 +28,7 @@
>  #include <linux/slab.h>
>  #include <linux/delay.h>
>  #include <linux/smp_lock.h>
> +#include <linux/kthread.h>
>  #include <asm/div64.h>
>
>  #include "saa7134-reg.h"
> @@ -357,7 +358,7 @@ static int tvaudio_sleep(struct saa7134_
>   DECLARE_WAITQUEUE(wait, current);
>
>   add_wait_queue(&dev->thread.wq, &wait);
> - if (dev->thread.scan1 == dev->thread.scan2 && !dev->thread.shutdown) {
> + if (dev->thread.scan1 == dev->thread.scan2 && !kthread_should_stop()) {
>    if (timeout < 0) {
>     set_current_state(TASK_INTERRUPTIBLE);
>     schedule();
> @@ -525,7 +526,7 @@ static int tvaudio_thread(void *data)
>   allow_signal(SIGTERM);
>   for (;;) {
>    tvaudio_sleep(dev,-1);
> -  if (dev->thread.shutdown || signal_pending(current))
> +  if (kthread_should_stop() || signal_pending(current))
>     goto done;
>
>   restart:
> @@ -633,7 +634,7 @@ static int tvaudio_thread(void *data)
>    for (;;) {
>     if (tvaudio_sleep(dev,5000))
>      goto restart;
> -   if (dev->thread.shutdown || signal_pending(current))
> +   if (kthread_should_stop() || signal_pending(current))
>      break;
>     if (UNSET == dev->thread.mode) {
>      rx = tvaudio_getstereo(dev,&tvaudio[i]);
> @@ -649,7 +650,6 @@ static int tvaudio_thread(void *data)
>   }
>
>   done:
> - complete_and_exit(&dev->thread.exit, 0);
>   return 0;
>  }
>
> @@ -798,7 +798,6 @@ static int tvaudio_thread_ddep(void *dat
>   struct saa7134_dev *dev = data;
>   u32 value, norms, clock;
>
> - daemonize("%s", dev->name);
>   allow_signal(SIGTERM);
>
>   clock = saa7134_boards[dev->board].audio_clock;
> @@ -812,7 +811,7 @@ static int tvaudio_thread_ddep(void *dat
>
>   for (;;) {
>    tvaudio_sleep(dev,-1);
> -  if (dev->thread.shutdown || signal_pending(current))
> +  if (kthread_should_stop() || signal_pending(current))
>     goto done;
>
>   restart:
> @@ -894,7 +893,6 @@ static int tvaudio_thread_ddep(void *dat
>   }
>
>   done:
> - complete_and_exit(&dev->thread.exit, 0);
>   return 0;
>  }
>
> @@ -1004,15 +1002,16 @@ int saa7134_tvaudio_init2(struct saa7134
>    break;
>   }
>
> - dev->thread.pid = -1;
> + dev->thread.task = NULL;
>   if (my_thread) {
>    /* start tvaudio thread */
>    init_waitqueue_head(&dev->thread.wq);
> -  init_completion(&dev->thread.exit);
> -  dev->thread.pid = kernel_thread(my_thread,dev,0);
> -  if (dev->thread.pid < 0)
> +  dev->thread.task = kthread_run(my_thread,dev,dev->name);
> +  if (IS_ERR(dev->thread.task)) {
>     printk(KERN_WARNING "%s: kernel_thread() failed\n",
> -          dev->name);
> +                                dev->name);
> +   dev->thread.task = NULL;
> +  }
>    saa7134_tvaudio_do_scan(dev);
>   }
>
> @@ -1023,10 +1022,10 @@ int saa7134_tvaudio_init2(struct saa7134
>  int saa7134_tvaudio_fini(struct saa7134_dev *dev)
>  {
>   /* shutdown tvaudio thread */
> - if (dev->thread.pid >= 0) {
> -  dev->thread.shutdown = 1;
> -  wake_up_interruptible(&dev->thread.wq);
> -  wait_for_completion(&dev->thread.exit);
> + if (dev->thread.task) {
> +  /* kthread_stop() wakes up the thread */
> +  kthread_stop(dev->thread.task);
> +  dev->thread.task = NULL;
>   }
>   saa_andorb(SAA7134_ANALOG_IO_SELECT, 0x07, 0x00); /* LINE1 */
>   return 0;
> @@ -1034,7 +1033,7 @@ int saa7134_tvaudio_fini(struct saa7134_
>
>  int saa7134_tvaudio_do_scan(struct saa7134_dev *dev)
>  {
> - if (dev->thread.pid >= 0) {
> + if (dev->thread.task) {
>    dev->thread.mode = UNSET;
>    dev->thread.scan2++;
>    wake_up_interruptible(&dev->thread.wq);
```

---

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
## Posted by ebiederm on Tue, 29 Aug 2006 22:39:53 GMT

View Forum Message <> Reply to Message

Andrew Morton <akpm@osdl.org> writes:

> So in general, yes, the driver should be converted to the kthread API -
> this is a requirement for virtualisation, but I forget why, and that's the
> "standard" way of doing it.

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.

Eric

---

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
## Posted by ebiederm on Wed, 30 Aug 2006 12:39:49 GMT

View Forum Message <> Reply to Message

ebiederm@xmission.com (Eric W. Biederman) writes:

> Andrew Morton <akpm@osdl.org> writes:
>
>> So in general, yes, the driver should be converted to the kthread API -
>> this is a requirement for virtualisation, but I forget why, and that's the
>> "standard" way of doing it.

>
> 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.

Eric

---

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
Posted by Cedric Le Goater on Wed, 30 Aug 2006 14:07:00 GMT

View Forum Message <> Reply to Message

>> 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.

thanks,

C.

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

## 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

---

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
## Posted by Cedric Le Goater on Wed, 30 Aug 2006 16:18:55 GMT

View Forum Message <> Reply to Message

Eric W. Biederman wrote:

[ ... ]

>>> 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.

I've extracted this list from a table which includes a pid cache column.
this pid cache column is not complete yet. I'd be nice if we could use a
wiki to maintain this table, the existing openvz or vserver wiki ?

> 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.

ok we need to add this one.

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

got that also.

> 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 :)

sure :) some drivers are for some exotic piece of hardware that are not
currently found on a standard server.

> 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.

ok thanks, will update.

> 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.

yes yes. hard stuff though which requires 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.

we would need a wiki to maintain the work in progress on that topic while
we work on the pidspace.

another list to maintain would be the pid_t to struct pid replacement.

C.

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
## Posted by Cedric Le Goater on Wed, 30 Aug 2006 16:30:27 GMT

View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Tue, 29 Aug 2006 14:15:55 -0700
> Sukadev Bhattiprolu <sukadev@us.ibm.com> wrote:
>
>> Replace kernel_thread() with kthread_run() since kernel_thread()
>> is deprecated in drivers/modules.
>>
>> Note that this driver, like a few others, allows SIGTERM. Not
>> sure if that is affected by conversion to kthread. Appreciate
>> any comments on that.
>>
>
> hm, I think this driver needs more help.
>
> - It shouldn't be using signals at all, really.  Signals are for
>   userspace IPC.  The kernel internally has better/richer/faster/tighter
>   ways of inter-thread communication.
>
> - saa7134_tvaudio_fini()-versus-tvaudio_sleep() looks racy:
>

>  if (dev->thread.scan1 == dev->thread.scan2 && !dev->thread.shutdown) {
>   if (timeout < 0) {
>    set_current_state(TASK_INTERRUPTIBLE);
>    schedule();
>
>   If the wakeup happens after the test of dev->thread.shutdown, that sleep will
>   be permanent.
>
>
> So in general, yes, the driver should be converted to the kthread API -
> this is a requirement for virtualisation, but I forget why, and that's the
> "standard" way of doing it.
>
> - The signal stuff should go away if at all possible.

The thread of this driver allows SIGTERM for some obscure reason. Not sure
why, I didn't find anything relying on it.

could we just remove the allow_signal() ?

C.

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
Posted by Andrew Morton on Wed, 30 Aug 2006 16:49:43 GMT
View Forum Message <> Reply to Message

On Wed, 30 Aug 2006 18:30:27 +0200
Cedric Le Goater <clg@fr.ibm.com> wrote:

> Andrew Morton wrote:
> > On Tue, 29 Aug 2006 14:15:55 -0700
> > Sukadev Bhattiprolu <sukadev@us.ibm.com> wrote:
> >
> >> Replace kernel_thread() with kthread_run() since kernel_thread()
> >> is deprecated in drivers/modules.
> >>
> >> Note that this driver, like a few others, allows SIGTERM. Not
> >> sure if that is affected by conversion to kthread. Appreciate
> >> any comments on that.
> >>
> >
> > hm, I think this driver needs more help.
> >

> > - It shouldn't be using signals at all, really.  Signals are for
> >   userspace IPC.  The kernel internally has better/richer/faster/tighter
> >   ways of inter-thread communication.
> >
> > - saa7134_tvaudio_fini()-versus-tvaudio_sleep() looks racy:
> >
> >  if (dev->thread.scan1 == dev->thread.scan2 && !dev->thread.shutdown) {
> >   if (timeout < 0) {
> >    set_current_state(TASK_INTERRUPTIBLE);
> >    schedule();
> >
> >   If the wakeup happens after the test of dev->thread.shutdown, that sleep will
> >   be permanent.
> >
> >
> > So in general, yes, the driver should be converted to the kthread API -
> > this is a requirement for virtualisation, but I forget why, and that's the
> > "standard" way of doing it.
> >
> > - The signal stuff should go away if at all possible.
>
> The thread of this driver allows SIGTERM for some obscure reason. Not sure
> why, I didn't find anything relying on it.
>
> could we just remove the allow_signal() ?
>

I hope so.  However I have a bad feeling that the driver wants to accept
signals from userspace.  Hopefully Mauro & co will be able to clue us in.

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
Posted by Mauro Carvalho Chehab on Wed, 30 Aug 2006 17:36:41 GMT
View Forum Message <> Reply to Message

> On Wed, 30 Aug 2006 18:30:27 +0200
> Cedric Le Goater <clg@fr.ibm.com> wrote:
>

> > The thread of this driver allows SIGTERM for some obscure reason. Not sure
> > why, I didn't find anything relying on it.
> >
> > could we just remove the allow_signal() ?

> >
>
> I hope so.  However I have a bad feeling that the driver wants to accept
> signals from userspace.  Hopefully Mauro & co will be able to clue us in.

The history we have on our development tree goes only until Feb, 2004.
This line were added before it.

I've looked briefly the driver. The same allow_signal is also present on
tvaudio (part of bttv driver). BTTV were written to kernel 2.1, so, this
piece seems to be an inheritance from 2.1 time.

No other V4L drivers have this one, although cx88-tvaudio (written on
2.6 series) have a similar kthread to check if audio status changed.

On cx88-tvaudio, it does:
```
		if (kthread_should_stop())
			break;
```
instead of:
```

		if (dev->thread.shutdown || signal_pending(current))
			goto done;
```

It is likely to work if we remove allow_signal and replacing the
signal_pending() by kthread_should_stop() as above.

The better is to check the real patch on a saa7134 hardware before
submiting to mainstream. You may submit the final patch for us to test
at the proper hardware.

Cheers,
Mauro.

## Subject: Re: [PATCH] kthread: saa7134-tvaudio.c
## Posted by Sukadev Bhattiprolu on Thu, 31 Aug 2006 01:02:49 GMT

View Forum Message <> Reply to Message

Mauro Carvalho Chehab [mchehab@infradead.org] wrote:

| > On Wed, 30 Aug 2006 18:30:27 +0200
| > Cedric Le Goater <clg@fr.ibm.com> wrote:
| >

|
| It is likely to work if we remove allow_signal and replacing the
| signal_pending() by kthread_should_stop() as above.
|
| The better is to check the real patch on a saa7134 hardware before
| submiting to mainstream. You may submit the final patch for us to test
| at the proper hardware.
|

Thanks for all the input. Mauro, thanks for help with testing.
Here is an updated patch that removes the signal code and the race.

---

Replace kernel_thread() with kthread_run() since kernel_thread()
is deprecated in drivers/modules. Also remove signalling code
as it is not needed in the driver.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Mauro Carvalho Chehab <mchehab@infradead.org>
Cc: Containers@lists.osdl.org
Cc: video4linux-list@redhat.com
Cc: v4l-dvb-maintainer@linuxtv.org

```
 drivers/media/video/saa7134/saa7134-tvaudio.c |   45 ++++++++++++-------------
 drivers/media/video/saa7134/saa7134.h         |    4 --
 2 files changed, 24 insertions(+), 25 deletions(-)

Index: lx26-18-rc5/drivers/media/video/saa7134/saa7134.h
===================================================================
--- lx26-18-rc5.orig/drivers/media/video/saa7134/saa7134.h 2006-08-29 18:35:53.000000000
-0700
+++ lx26-18-rc5/drivers/media/video/saa7134/saa7134.h 2006-08-29 18:35:56.000000000 -0700
@@ -311,10 +311,8 @@ struct saa7134_pgtable {

 /* tvaudio thread status */
 struct saa7134_thread {
-	pid_t                      pid;
-	struct completion          exit;
+	struct task_struct *       task;
 	wait_queue_head_t          wq;
-	unsigned int               shutdown;
 	unsigned int               scan1;
 	unsigned int               scan2;
 	unsigned int               mode;
```

```
Index: lx26-18-rc5/drivers/media/video/saa7134/saa7134-tvaudio.c
===================================================================
--- lx26-18-rc5.orig/drivers/media/video/saa7134/saa7134-tvaudio.c 2006-08-29
18:35:53.000000000 -0700
+++ lx26-18-rc5/drivers/media/video/saa7134/saa7134-tvaudio.c 2006-08-30 14:09:00.000000000
-0700
@@ -28,6 +28,7 @@
 #include <linux/slab.h>
 #include <linux/delay.h>
 #include <linux/smp_lock.h>
+#include <linux/kthread.h>
 #include <asm/div64.h>

 #include "saa7134-reg.h"
@@ -357,16 +358,22 @@ static int tvaudio_sleep(struct saa7134_
  DECLARE_WAITQUEUE(wait, current);

  add_wait_queue(&dev->thread.wq, &wait);
- if (dev->thread.scan1 == dev->thread.scan2 && !dev->thread.shutdown) {
+
+ set_current_state(TASK_INTERRUPTIBLE);
+
+ if (dev->thread.scan1 == dev->thread.scan2 && !kthread_should_stop()) {
   if (timeout < 0) {
-   set_current_state(TASK_INTERRUPTIBLE);
    schedule();
   } else {
    schedule_timeout_interruptible
       (msecs_to_jiffies(timeout));
   }
  }
+
+ set_current_state(TASK_RUNNING);
+
  remove_wait_queue(&dev->thread.wq, &wait);
+
  return dev->thread.scan1 != dev->thread.scan2;
 }

@@ -521,11 +528,9 @@ static int tvaudio_thread(void *data)
  unsigned int i, audio, nscan;
  int max1,max2,carrier,rx,mode,lastmode,default_carrier;

- daemonize("%s", dev->name);
- allow_signal(SIGTERM);
  for (;;) {
   tvaudio_sleep(dev,-1);
-  if (dev->thread.shutdown || signal_pending(current))
```

```
+  if (kthread_should_stop())
    goto done;

  restart:
@@ -633,7 +638,7 @@ static int tvaudio_thread(void *data)
   for (;;) {
    if (tvaudio_sleep(dev,5000))
     goto restart;
-   if (dev->thread.shutdown || signal_pending(current))
+   if (kthread_should_stop())
     break;
    if (UNSET == dev->thread.mode) {
     rx = tvaudio_getstereo(dev,&tvaudio[i]);
@@ -649,7 +654,6 @@ static int tvaudio_thread(void *data)
  }

  done:
- complete_and_exit(&dev->thread.exit, 0);
  return 0;
 }

@@ -798,9 +802,6 @@ static int tvaudio_thread_ddep(void *dat
  struct saa7134_dev *dev = data;
  u32 value, norms, clock;

- daemonize("%s", dev->name);
- allow_signal(SIGTERM);
-
  clock = saa7134_boards[dev->board].audio_clock;
  if (UNSET != audio_clock_override)
   clock = audio_clock_override;
@@ -812,7 +813,7 @@ static int tvaudio_thread_ddep(void *dat

  for (;;) {
   tvaudio_sleep(dev,-1);
-  if (dev->thread.shutdown || signal_pending(current))
+  if (kthread_should_stop())
    goto done;

  restart:
@@ -894,7 +895,6 @@ static int tvaudio_thread_ddep(void *dat
  }

  done:
- complete_and_exit(&dev->thread.exit, 0);
  return 0;
 }
```

```
@@ -1004,15 +1004,16 @@ int saa7134_tvaudio_init2(struct saa7134
   break;
  }

- dev->thread.pid = -1;
+ dev->thread.task = NULL;
  if (my_thread) {
   /* start tvaudio thread */
   init_waitqueue_head(&dev->thread.wq);
-  init_completion(&dev->thread.exit);
-  dev->thread.pid = kernel_thread(my_thread,dev,0);
-  if (dev->thread.pid < 0)
-   printk(KERN_WARNING "%s: kernel_thread() failed\n",
+  dev->thread.task = kthread_run(my_thread, dev, dev->name);
+  if (IS_ERR(dev->thread.task)) {
+   printk(KERN_WARNING "%s: failed to create kthread\n",
          dev->name);
+   dev->thread.task = NULL;
+  }
   saa7134_tvaudio_do_scan(dev);
  }

@@ -1023,10 +1024,10 @@ int saa7134_tvaudio_init2(struct saa7134
 int saa7134_tvaudio_fini(struct saa7134_dev *dev)
 {
  /* shutdown tvaudio thread */
- if (dev->thread.pid >= 0) {
-  dev->thread.shutdown = 1;
-  wake_up_interruptible(&dev->thread.wq);
-  wait_for_completion(&dev->thread.exit);
+ if (dev->thread.task) {
+  /* kthread_stop() wakes up the thread */
+  kthread_stop(dev->thread.task);
+  dev->thread.task = NULL;
  }
  saa_andorb(SAA7134_ANALOG_IO_SELECT, 0x07, 0x00); /* LINE1 */
  return 0;
@@ -1034,7 +1035,7 @@ int saa7134_tvaudio_fini(struct saa7134_

 int saa7134_tvaudio_do_scan(struct saa7134_dev *dev)
 {
- if (dev->thread.pid >= 0) {
+ if (dev->thread.task) {
   dev->thread.mode = UNSET;
   dev->thread.scan2++;
   wake_up_interruptible(&dev->thread.wq);
```

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

Subject: [PATCH] kthread: tvaudio.c
Posted by Sukadev Bhattiprolu on Thu, 31 Aug 2006 01:05:04 GMT
View Forum Message <> Reply to Message

Replaced kernel_thread() with kthread_run() since kernel_thread() is
deprecated in drivers/modules.

Removed the completion and the wait queue which are now useless with
kthread. Also removed the allow_signal() call as signals don't apply
to kernel threads.

Fixed a small race condition when thread is stopped.

Please check if the timer vs. thread still works fine without the wait
queue.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Cc: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: Mauro Carvalho Chehab <mchehab@infradead.org>
Cc: Containers@lists.osdl.org
Cc: video4linux-list@redhat.com
Cc: v4l-dvb-maintainer@linuxtv.org

```
 drivers/media/video/tvaudio.c |   42 ++++++++++++++++--------------------------
 1 files changed, 16 insertions(+), 26 deletions(-)

Index: lx26-18-rc5/drivers/media/video/tvaudio.c
===================================================================
--- lx26-18-rc5.orig/drivers/media/video/tvaudio.c 2006-08-29 14:02:44.000000000 -0700
+++ lx26-18-rc5/drivers/media/video/tvaudio.c 2006-08-30 17:52:17.000000000 -0700
@@ -28,6 +28,7 @@
 #include <linux/i2c-algo-bit.h>
 #include <linux/init.h>
 #include <linux/smp_lock.h>
+#include <linux/kthread.h>

 #include <media/tvaudio.h>
 #include <media/v4l2-common.h>
@@ -124,11 +125,8 @@ struct CHIPSTATE {
  int input;
  /* thread */
- pid_t                tpid;
- struct completion    texit;
- wait_queue_head_t    wq;
+ struct task_struct   *thread;
  struct timer_list    wt;
- int                  done;
  int                  watch_stereo;
  int        audmode;
 };
@@ -264,28 +262,23 @@ static int chip_cmd(struct CHIPSTATE *ch
 static void chip_thread_wake(unsigned long data)
 {
  struct CHIPSTATE *chip = (struct CHIPSTATE*)data;
- wake_up_interruptible(&chip->wq);
+ wake_up_process(chip->thread);
 }

 static int chip_thread(void *data)
 {
- DECLARE_WAITQUEUE(wait, current);
  struct CHIPSTATE *chip = data;
  struct CHIPDESC  *desc = chiplist + chip->type;

- daemonize("%s", chip->c.name);
- allow_signal(SIGTERM);
  v4l_dbg(1, debug, &chip->c, "%s: thread started\n", chip->c.name);

  for (;;) {
-  add_wait_queue(&chip->wq, &wait);
-  if (!chip->done) {
-   set_current_state(TASK_INTERRUPTIBLE);
+  set_current_state(TASK_INTERRUPTIBLE);
+  if (!kthread_should_stop())
    schedule();
-  }
-  remove_wait_queue(&chip->wq, &wait);
+  set_current_state(TASK_RUNNING);
   try_to_freeze();
-  if (chip->done || signal_pending(current))
+  if (kthread_should_stop())
    break;
   v4l_dbg(1, debug, &chip->c, "%s: thread wakeup\n", chip->c.name);

@@ -301,7 +294,6 @@ static int chip_thread(void *data)
  }

  v4l_dbg(1, debug, &chip->c, "%s: thread exiting\n", chip->c.name);
```

```
- complete_and_exit(&chip->texit, 0);
  return 0;
 }

@@ -1536,19 +1528,18 @@ static int chip_attach(struct i2c_adapte
   chip_write(chip,desc->treblereg,desc->treblefunc(chip->treble));
  }

- chip->tpid = -1;
+ chip->thread = NULL;
  if (desc->checkmode) {
   /* start async thread */
   init_timer(&chip->wt);
   chip->wt.function = chip_thread_wake;
   chip->wt.data     = (unsigned long)chip;
-  init_waitqueue_head(&chip->wq);
-  init_completion(&chip->texit);
-  chip->tpid = kernel_thread(chip_thread,(void *)chip,0);
-  if (chip->tpid < 0)
-   v4l_warn(&chip->c, "%s: kernel_thread() failed\n",
+  chip->thread = kthread_run(chip_thread, chip, chip->c.name);
+  if (IS_ERR(chip->thread)) {
+   v4l_warn(&chip->c, "%s: failed to create kthread\n",
           chip->c.name);
-  wake_up_interruptible(&chip->wq);
+   chip->thread = NULL;
+  }
  }
  return 0;
 }
@@ -1569,11 +1560,10 @@ static int chip_detach(struct i2c_client
  struct CHIPSTATE *chip = i2c_get_clientdata(client);

  del_timer_sync(&chip->wt);
- if (chip->tpid >= 0) {
+ if (chip->thread) {
   /* shutdown async thread */
-  chip->done = 1;
-  wake_up_interruptible(&chip->wq);
-  wait_for_completion(&chip->texit);
+  kthread_stop(chip->thread);
+  chip->thread = NULL;
  }

  i2c_detach_client(&chip->c);
```

https://lists.osdl.org/mailman/listinfo/containers