Subject: Re: [PATCH] usbatm: Update to use the kthread api.
Posted by Cedric Le Goater on Wed, 13 Dec 2006 16:24:17 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> During driver initialization if the driver has an expensive
> initialization routine usbatm starts a separate kernel thread for it.
>
> In the driver cleanup routine the code waits to ensure the
> initialization routine has finished.
>
> Switching to the kthread api allowed some of the thread management
> code to be removed.
>
> In addition the kill_proc(SIGTERM, ...) in usbatm_usb_disconnect was
> removed because it was absolutely pointless.  The kernel thread did
> not handle SIGTERM or any pending signals, so despite marking the
> signal as pending it would never have been handled.

are you sure that the heavy_init() routines don't handle pending
signals. they do firmware loading, etc. ?

> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
> ---
>  drivers/usb/atm/usbatm.c |   24 ++++++----------------
>  drivers/usb/atm/usbatm.h |    2 --
>  2 files changed, 6 insertions(+), 20 deletions(-)
>
> diff --git a/drivers/usb/atm/usbatm.c b/drivers/usb/atm/usbatm.c
> index ec63b0e..e6cd5e4 100644
> --- a/drivers/usb/atm/usbatm.c
> +++ b/drivers/usb/atm/usbatm.c
> @@ -81,6 +81,7 @@
>  #include <linux/stat.h>
>  #include <linux/timer.h>
>  #include <linux/wait.h>
> +#include <linux/kthread.h>
>
>  #ifdef VERBOSE_DEBUG
>  static int usbatm_print_packet(const unsigned char *data, int len);
> @@ -999,35 +1000,26 @@ static int usbatm_do_heavy_init(void *arg)
>   struct usbatm_data *instance = arg;
>   int ret;
>
> - daemonize(instance->driver->driver_name);
> - allow_signal(SIGTERM);
> - instance->thread_pid = current->pid;
> -

```
> - complete(&instance->thread_started);
> -
>   ret = instance->driver->heavy_init(instance, instance->usb_intf);
>
>   if (!ret)
>    ret = usbatm_atm_init(instance);
>
> - mutex_lock(&instance->serialize);
> - instance->thread_pid = -1;
> - mutex_unlock(&instance->serialize);
>
>   complete_and_exit(&instance->thread_exited, ret);
>  }
>
>  static int usbatm_heavy_init(struct usbatm_data *instance)
>  {
> - int ret = kernel_thread(usbatm_do_heavy_init, instance, CLONE_KERNEL);
> -
> - if (ret < 0) {
> + struct task_struct *thread;
> + thread = kthread_run(usbatm_do_heavy_init, instance,
> +     instance->driver->driver_name);
> + if (IS_ERR(thread)) {
> +  int ret = PTR_ERR(thread);
>    usb_err(instance, "%s: failed to create kernel_thread (%d)!\n", __func__, ret);
>    return ret;
>   }
>
> - wait_for_completion(&instance->thread_started);
> -
>   return 0;
>  }
>
> @@ -1109,8 +1101,6 @@ int usbatm_usb_probe(struct usb_interface *intf, const struct
usb_device_id *id,
>   kref_init(&instance->refcount);  /* dropped in usbatm_usb_disconnect */
>   mutex_init(&instance->serialize);
>
> - instance->thread_pid = -1;
> - init_completion(&instance->thread_started);
>   init_completion(&instance->thread_exited);
>
>   INIT_LIST_HEAD(&instance->vcc_list);
> @@ -1272,8 +1262,6 @@ void usbatm_usb_disconnect(struct usb_interface *intf)
>
>   mutex_lock(&instance->serialize);
>   instance->disconnected = 1;
> - if (instance->thread_pid >= 0)
```

> -  kill_proc(instance->thread_pid, SIGTERM, 1);
>   mutex_unlock(&instance->serialize);
>
>   wait_for_completion(&instance->thread_exited);
> diff --git a/drivers/usb/atm/usbatm.h b/drivers/usb/atm/usbatm.h
> index ff8551e..ab42355 100644
> --- a/drivers/usb/atm/usbatm.h
> +++ b/drivers/usb/atm/usbatm.h
> @@ -176,8 +176,6 @@ struct usbatm_data {
>   int disconnected;
>
>   /* heavy init */
> - int thread_pid;
> - struct completion thread_started;
>   struct completion thread_exited;
>
>   /* ATM device */

_____

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