
Subject: Re: [PATCH] bluetooth rfcomm: Convert to kthread API.

Posted by [akpm](#) on Thu, 19 Apr 2007 23:12:53 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Thu, 19 Apr 2007 01:58:54 -0600

"Eric W. Biederman" <ebiederm@xmission.com> wrote:

> From: Eric W. Biederman <ebiederm@xmission.com>

>

> This patch starts krfcommd using kthread_run instead of a combination

> of kernel_thread and daemonize making the code slightly simpler

> and more maintainable.

gargh, the more I look at these things, the more I agree with Christoph.

> Cc: Marcel Holtmann <marcel@holtmann.org>

> Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

> ---

> net/bluetooth/rfcomm/core.c | 4 +---

> 1 files changed, 2 insertions(+), 2 deletions(-)

>

> diff --git a/net/bluetooth/rfcomm/core.c b/net/bluetooth/rfcomm/core.c

> index 34f993a..baaad49 100644

> --- a/net/bluetooth/rfcomm/core.c

> +++ b/net/bluetooth/rfcomm/core.c

> @@ -38,6 +38,7 @@

> #include <linux/net.h>

> #include <linux/mutex.h>

> #include <linux/freezer.h>

> +#include <linux/kthread.h>

>

> #include <net/sock.h>

> #include <asm/uaccess.h>

> @@ -1938,7 +1939,6 @@ static int rfcomm_run(void *unused)

>

> atomic_inc(&running);

>

> - daemonize("krfcommd");

> set_user_nice(current, -10);

>

> BT_DBG("");

> @@ -2058,7 +2058,7 @@ static int __init rfcomm_init(void)

>

> hci_register_cb(&rfcomm_cb);

>

> - kernel_thread(rfcomm_run, NULL, CLONE_KERNEL);

> + kthread_run(rfcomm_run, NULL, "krfcommd");

>

```
> if (class_create_file(bt_class, &class_attr_rfcomm_dlc) < 0)
>   BT_ERR("Failed to create RFCOMM info file");
```

We should remove the file-wide `terminate' and `running' and switch the thread management over to kthread_run(), kthread_stop() and kthread_should_stop().

btw, this:

```
static void rfcomm_worker(void)
{
    BT_DBG("");

    while (!atomic_read(&terminate)) {
        try_to_freeze();

        if (!test_bit(RFCOMM_SCHED_WAKEUP, &rfcomm_event)) {
            /* No pending events. Let's sleep.
             * Incoming connections and data will wake us up. */
            set_current_state(TASK_INTERRUPTIBLE);
            schedule();
        }

        /* Process stuff */
        clear_bit(RFCOMM_SCHED_WAKEUP, &rfcomm_event);
        rfcomm_process_sessions();
    }
    set_current_state(TASK_RUNNING);
    return;
}
```

appears to have the classic sleep/wakeup bug: if the wakeup happens after we tested RFCOMM_SCHED_WAKEUP we will miss it.

Easy fix:

From: Andrew Morton <akpm@linux-foundation.org>

Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

```
net/bluetooth/rfcomm/core.c | 4 ++--
1 files changed, 2 insertions(+), 2 deletions(-)
```

```
diff -puN net/bluetooth/rfcomm/core.c~rfcomm_worker-fix-wakeup-race
net/bluetooth/rfcomm/core.c
--- a/net/bluetooth/rfcomm/core.c~rfcomm_worker-fix-wakeup-race
+++ a/net/bluetooth/rfcomm/core.c
```

```

@@ -1855,18 +1855,18 @@ static void rfcomm_worker(void)
    while (!atomic_read(&terminate)) {
        try_to_freeze();

+   set_current_state(TASK_INTERRUPTIBLE);
        if (!test_bit(RFCOMM_SCHED_WAKEUP, &rfcomm_event)) {
            /* No pending events. Let's sleep.
             * Incoming connections and data will wake us up. */
-   set_current_state(TASK_INTERRUPTIBLE);
            schedule();
        }
+   set_current_state(TASK_RUNNING);

        /* Process stuff */
        clear_bit(RFCOMM_SCHED_WAKEUP, &rfcomm_event);
        rfcomm_process_sessions();
    }
- set_current_state(TASK_RUNNING);
    return;
}

```

—

(I think it's safer and saner to always run rfcomm_process_sessions() while in state TASK_RUNNING, not maybe-in-state-TASK_INTERRUPTIBLE)

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