Subject: Re: Re: [PATCH] bluetooth rfcomm: Convert to kthread API.
Posted by Cedric Le Goater on Fri, 20 Apr 2007 15:20:16 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> 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().

here's an old refreshed one using kthread_stop() and fixing the racy
wakeup.

C.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

---
 net/bluetooth/rfcomm/core.c | 61 ++++++++++++++++++++----------------------
 1 file changed, 21 insertions(+), 40 deletions(-)

Index: 2.6.21-rc6-mm1/net/bluetooth/rfcomm/core.c
 == =======
--- 2.6.21-rc6-mm1.orig/net/bluetooth/rfcomm/core.c
+++ 2.6.21-rc6-mm1/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>
@@ -68,7 +69,6 @@ static DEFINE_MUTEX(rfcomm_mutex);
 static unsigned long rfcomm_event;

 static LIST_HEAD(session_list);
-static atomic_t terminate, running;

 static int rfcomm_send_frame(struct rfcomm_session *s, u8 *data, int len);
 static int rfcomm_send_sabm(struct rfcomm_session *s, u8 dlci);
@@ -1847,28 +1847,6 @@ static inline void rfcomm_process_sessio
  rfcomm_unlock();
 }

-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;
-}
-
 static int rfcomm_add_listener(bdaddr_t *ba)
 {
  struct sockaddr_l2 addr;
@@ -1934,22 +1912,29 @@ static void rfcomm_kill_listener(void)

 static int rfcomm_run(void *unused)
 {
- rfcomm_thread = current;
-
- atomic_inc(&running);
-
- daemonize("krfcommd");
  set_user_nice(current, -10);

  BT_DBG("");

  rfcomm_add_listener(BDADDR_ANY);

- rfcomm_worker();
+ while (!kthread_should_stop()) {
+  try_to_freeze();

- rfcomm_kill_listener();
+  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. */
+   schedule();
+  }
```

```
- atomic_dec(&running);
+  /* Process stuff */
+  clear_bit(RFCOMM_SCHED_WAKEUP, &rfcomm_event);
+  rfcomm_process_sessions();
+ }
+ set_current_state(TASK_RUNNING);
+
+ rfcomm_kill_listener();
  return 0;
 }

@@ -2058,7 +2043,9 @@ static int __init rfcomm_init(void)

  hci_register_cb(&rfcomm_cb);

- kernel_thread(rfcomm_run, NULL, CLONE_KERNEL);
+ rfcomm_thread = kthread_run(rfcomm_run, NULL, "krfcommd");
+ if (IS_ERR(rfcomm_thread))
+  return PTR_ERR(rfcomm_thread);

  if (class_create_file(bt_class, &class_attr_rfcomm_dlc) < 0)
   BT_ERR("Failed to create RFCOMM info file");
@@ -2080,14 +2067,8 @@ static void __exit rfcomm_exit(void)

  hci_unregister_cb(&rfcomm_cb);

- /* Terminate working thread.
-  * ie. Set terminate flag and wake it up */
- atomic_inc(&terminate);
- rfcomm_schedule(RFCOMM_SCHED_STATE);
-
- /* Wait until thread is running */
- while (atomic_read(&running))
-  schedule();
+ /* Terminate working thread */
+ kthread_stop(rfcomm_thread);

 #ifdef CONFIG_BT_RFCOMM_TTY
  rfcomm_cleanup_ttys();
```