
Subject: Re: Remaining straight forward kthread API conversions...
Posted by [Christoph Hellwig](#) on Sun, 22 Apr 2007 12:15:24 GMT
[View Forum Message](#) <> [Reply to Message](#)

Looks like you were missing at least the pcie hotplug driver. Another one of the horrible thread abuses in drivers/pci/hotplug.

- full conversion to kthread infrastructure
- use wake_up_process to wake the thread up

Like most pci hotplug drivers it still uses very race non-atomic variable assignment to communicated with the thread, but that's something the maintainers should look into.

Signed-off-by: Christoph Hellwig <hch@lst.de>

Index: linux-2.6/drivers/pci/hotplug/pciehp_ctrl.c

```
=====
--- linux-2.6.orig/drivers/pci/hotplug/pciehp_ctrl.c 2007-04-22 11:36:58.000000000 +0200
+++ linux-2.6/drivers/pci/hotplug/pciehp_ctrl.c 2007-04-22 11:42:56.000000000 +0200
```

```
@ @ -32,14 +32,13 @ @
```

```
#include <linux/types.h>
```

```
#include <linux/smp_lock.h>
```

```
#include <linux/pci.h>
```

```
+#include <linux/kthread.h>
```

```
#include "../pci.h"
```

```
#include "pciehp.h"
```

```
static void interrupt_event_handler(struct controller *ctrl);
```

```
-static struct semaphore event_semaphore; /* mutex for process loop (up if something to process) */
```

```
-static struct semaphore event_exit; /* guard ensure thread has exited before calling it quits */
```

```
-static int event_finished;
```

```
+static struct task_struct *pciehp_event_thread;
```

```
static unsigned long pushbutton_pending; /* = 0 */
```

```
static unsigned long surprise_rm_pending; /* = 0 */
```

```
@ @ -93,8 +92,9 @ @ u8 pciehp_handle_attention_button(u8 hp_
    info("Button ignore on Slot(%s)\n", slot_name(p_slot));
}
```

```
+ /* signal event thread that new event is posted */
if (rc)
```

```
- up(&event_semaphore); /* signal event thread that new event is posted */
```

```
+ wake_up_process(pciehp_event_thread);
```

```

return 0;

@@ -135,8 +135,9 @@ u8 pciehp_handle_switch_change(u8 hp_slo
    taskInfo->event_type = INT_SWITCH_CLOSE;
}

+ /* signal event thread that new event is posted */
+ if (rc)
+   up(&event_semaphore); /* signal event thread that new event is posted */
+ wake_up_process(pciehpd_event_thread);

return rc;
}
@@ -178,8 +179,9 @@ u8 pciehp_handle_presence_change(u8 hp_s
    taskInfo->event_type = INT_PRESENCE_OFF;
}

+ /* signal event thread that new event is posted */
+ if (rc)
+   up(&event_semaphore); /* signal event thread that new event is posted */
+ wake_up_process(pciehpd_event_thread);

return rc;
}
@@ -217,8 +219,10 @@ u8 pciehp_handle_power_fault(u8 hp_slot,
    taskInfo->event_type = INT_POWER_FAULT;
    info("power fault bit %x set\n", hp_slot);
}
+
+ /* signal event thread that new event is posted */
+ if (rc)
+   up(&event_semaphore); /* signal event thread that new event is posted */
+ wake_up_process(pciehpd_event_thread);

return rc;
}
@@ -362,7 +366,7 @@ static void pushbutton_helper_thread(uns
{
    pushbutton_pending = data;

- up(&event_semaphore);
+ wake_up_process(pciehpd_event_thread);
}

/**
@@ -452,19 +456,14 @@ static void pciehp_surprise_rm_thread(un

```

```

/* this is the main worker thread */
-static int event_thread(void* data)
+static int event_thread(void *data)
{
    struct controller *ctrl;
- lock_kernel();
- daemonize("pciehp_event");
-
- unlock_kernel();

    while (1) {
- dbg("!!!!event_thread sleeping\n");
- down_interruptible (&event_semaphore);
- dbg("event_thread woken finished = %d\n", event_finished);
- if (event_finished || signal_pending(current))
+ set_current_state(TASK_INTERRUPTIBLE);
+ schedule();
+ if (kthread_should_stop())
        break;
        /* Do stuff here */
        if (pushbutton_pending)
@@ -476,24 +475,15 @@ static int event_thread(void* data)
        interrupt_event_handler(ctrl);
    }
    dbg("event_thread signals exit\n");
- up(&event_exit);
    return 0;
}

int pciehp_event_start_thread(void)
{
- int pid;
-
- /* initialize our semaphores */
- init_MUTEX_LOCKED(&event_exit);
- event_finished=0;
-
- init_MUTEX_LOCKED(&event_semaphore);
- pid = kernel_thread(event_thread, NULL, 0);
-
- if (pid < 0) {
+ pciehp_event_thread = kthread_run(event_thread, NULL, "pciehp_event");
+ if (IS_ERR(pciehp_event_thread)) {
        err ("Can't start up our event thread\n");
- return -1;
+ return PTR_ERR(pciehp_event_thread);
    }
    return 0;
}

```

```

}
@@ -501,9 +491,7 @@ int pciehp_event_start_thread(void)

void pciehp_event_stop_thread(void)
{
- event_finished = 1;
- up(&event_semaphore);
- down(&event_exit);
+ kthread_stop(pciehpd_event_thread);
}

@@ -624,7 +612,7 @@ static void interrupt_event_handler(stru
    dbg("Surprise Removal\n");
    if (p_slot) {
        surprise_rm_pending = (unsigned long) p_slot;
-        up(&event_semaphore);
+        wake_up_process(pciehpd_event_thread);
        update_slot_info(p_slot);
    }
}

```

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