Subject: Re: [PATCH] cpqphp: Convert to use the kthread API
Posted by Christoph Hellwig on Sun, 22 Apr 2007 12:12:08 GMT
View Forum Message <> Reply to Message

On Thu, Apr 19, 2007 at 12:55:31AM -0600, Eric W. Biederman wrote:
> From: Eric W. Biederman <ebiederm@xmission.com> - unquoted
>
> This patch changes cpqphp to use kthread_run and not
> kernel_thread and daemonize to startup and setup
> the cpqphp thread.

Thread handling in this driver (and actually everything else) seems
to be written by a crackmonkey.

Here's my take at fixing everything slightly related to thread handling
up:

 - full switch to kthread infrastructure
 - remove unused semaphore as mutex and waitqueue in long_delay -
   in fact that whole function should just go away as the user would
   be a lot more happy with just msleep_interruptible.
 - use wake_up_process for waking the thread

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

Index: linux-2.6/drivers/pci/hotplug/cpqphp_ctrl.c
===
--- linux-2.6.orig/drivers/pci/hotplug/cpqphp_ctrl.c 2007-04-22 12:46:33.000000000 +0200
+++ linux-2.6/drivers/pci/hotplug/cpqphp_ctrl.c 2007-04-22 12:53:58.000000000 +0200
@@ -37,6 +37,7 @@
 #include <linux/smp_lock.h>
 #include <linux/pci.h>
 #include <linux/pci_hotplug.h>
+#include <linux/kthread.h>
 #include "cpqphp.h"

 static u32 configure_new_device(struct controller* ctrl, struct pci_func *func,
@@ -45,34 +46,20 @@ static int configure_new_function(struct
    u8 behind_bridge, struct resource_lists *resources);
 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 unsigned long pushbutton_pending; /* = 0 */

```
-/* things needed for the long_delay function */
-static struct semaphore  delay_sem;
-static wait_queue_head_t delay_wait;
+static struct task_struct *cpqhp_event_thread;
+static unsigned long pushbutton_pending; /* = 0 */

 /* delay is in jiffies to wait for */
 static void long_delay(int delay)
 {
- DECLARE_WAITQUEUE(wait, current);
-
- /* only allow 1 customer into the delay queue at once
-  * yes this makes some people wait even longer, but who really cares?
-  * this is for _huge_ delays to make the hardware happy as the
-  * signals bounce around
+ /*
+  * XXX(hch): if someone is bored please convert all callers
+  * to call msleep_interruptible directly.  They really want
+  * to specify timeouts in natural units and spend a lot of
+  * effort converting them to jiffies..
   */
- down (&delay_sem);
-
- init_waitqueue_head(&delay_wait);
-
- add_wait_queue(&delay_wait, &wait);
  msleep_interruptible(jiffies_to_msecs(delay));
- remove_wait_queue(&delay_wait, &wait);
-
- up(&delay_sem);
 }


@@ -955,8 +942,8 @@ irqreturn_t cpqhp_ctrl_intr(int IRQ, voi
  }

  if (schedule_flag) {
-  up(&event_semaphore);
-  dbg("Signal event_semaphore\n");
+  wake_up_process(cpqhp_event_thread);
+  dbg("Waking even thread");
  }
  return IRQ_HANDLED;
 }
@@ -1738,7 +1725,7 @@ static u32 remove_board(struct pci_func
 static void pushbutton_helper_thread(unsigned long data)
 {
  pushbutton_pending = data;
```

```
- up(&event_semaphore);
+ wake_up_process(cpqhp_event_thread);
 }


@@ -1746,16 +1733,14 @@ static void pushbutton_helper_thread(uns
 static int event_thread(void* data)
 {
  struct controller *ctrl;
- lock_kernel();
- daemonize("phpd_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) break;
+  set_current_state(TASK_INTERRUPTIBLE);
+  schedule();
+
+  if (kthread_should_stop())
+   break;
   /* Do stuff here */
   if (pushbutton_pending)
    cpqhp_pushbutton_thread(pushbutton_pending);
@@ -1764,38 +1749,24 @@ static int event_thread(void* data)
    interrupt_event_handler(ctrl);
  }
  dbg("event_thread signals exit\n");
- up(&event_exit);
  return 0;
 }

-
 int cpqhp_event_start_thread(void)
 {
- int pid;
-
- /* initialize our semaphores */
- init_MUTEX(&delay_sem);
- init_MUTEX_LOCKED(&event_semaphore);
- init_MUTEX_LOCKED(&event_exit);
- event_finished=0;
-
- pid = kernel_thread(event_thread, NULL, 0);
- if (pid < 0) {
```

```
+ cpqhp_event_thread = kthread_run(event_thread, NULL, "phpd_event");
+ if (IS_ERR(cpqhp_event_thread)) {
   err ("Can't start up our event thread\n");
-  return -1;
+  return PTR_ERR(cpqhp_event_thread);
  }
- dbg("Our event thread pid = %d\n", pid);
+
  return 0;
 }


 void cpqhp_event_stop_thread(void)
 {
- event_finished = 1;
- dbg("event_thread finish command given\n");
- up(&event_semaphore);
- dbg("wait for event_thread to exit\n");
- down(&event_exit);
+ kthread_stop(cpqhp_event_thread);
 }
```

_____

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