

---

Subject: [PATCH 8/12] pid: Replace is\_orphaned\_pgrp with  
is\_current\_pgrp\_orphaned

Posted by [ebiederm](#) on Wed, 13 Dec 2006 11:07:52 GMT

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

---

Every call to is\_orphaned\_pgrp passed in process\_group(current)  
which is racy with respect to another thread changing our process  
group. It didn't bite us because we were dealing with integers  
and the worse we would get would be a stale answer.

In switching the checks to use struct pid to be a little more  
efficient and prepare the way for pid namespaces this  
race became apparent.

So I simplified the calls to the more specialized is\_current\_pgrp\_orphaned  
so I didn't have to worry about making logic changes to avoid the race.

Signed-off-by: Eric W. Biederman <[ebiederm@xmission.com](mailto:ebiederm@xmission.com)>

---

drivers/char/n\_tty.c | 2 +-  
drivers/char/tty\_io.c | 2 +-  
include/linux/tty.h | 2 +-  
kernel/exit.c | 4 ++--  
kernel/signal.c | 2 +-  
5 files changed, 6 insertions(+), 6 deletions(-)

diff --git a/drivers/char/n\_tty.c b/drivers/char/n\_tty.c

index e96a00f..7f1ded8 100644

--- a/drivers/char/n\_tty.c

+++ b/drivers/char/n\_tty.c

@@ -1189,7 +1189,7 @@ static int job\_control(struct tty\_struct \*tty, struct file \*file)

printk("read\_chan: tty->pgrp <= 0!\n");

else if (process\_group(current) != tty->pgrp) {

if (is\_ignored(SIGTTIN) ||

- is\_orphaned\_pgrp(process\_group(current)))

+ is\_current\_pgrp\_orphaned())

return -EIO;

kill\_pg(process\_group(current), SIGTTIN, 1);

return -ERESTARTSYS;

diff --git a/drivers/char/tty\_io.c b/drivers/char/tty\_io.c

index 74f078a..d8fdf45 100644

--- a/drivers/char/tty\_io.c

+++ b/drivers/char/tty\_io.c

@@ -1118,7 +1118,7 @@ int tty\_check\_change(struct tty\_struct \* tty)

return 0;

if (is\_ignored(SIGTTOU))

return 0;

- if (is\_orphaned\_pgrp(process\_group(current)))

```

+ if (is_current_pgrp_orphaned())
    return -EIO;
    (void) kill_pg(process_group(current), SIGTTOU, 1);
    return -ERESTARTSYS;
diff --git a/include/linux/tty.h b/include/linux/tty.h
index 1185bca..13a4918 100644
--- a/include/linux/tty.h
+++ b/include/linux/tty.h
@@ -283,7 +283,7 @@ extern int tty_read_raw_data(struct tty_struct *tty, unsigned char *bufp,
    int buflen);
extern void tty_write_message(struct tty_struct *tty, char *msg);

-extern int is_orphaned_pgrp(int pgrp);
+extern int is_current_pgrp_orphaned(void);
extern int is_ignored(int sig);
extern int tty_signal(int sig, struct tty_struct *tty);
extern void tty_hangup(struct tty_struct * tty);
diff --git a/kernel/exit.c b/kernel/exit.c
index 5f8455e..610917d 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -229,12 +229,12 @@ static int will_become_orphaned_pgrp(struct pid *pgrp, struct
task_struct *ignor
    return ret; /* (sighing) "Often!" */
}

-int is_orphaned_pgrp(int pgrp)
+int is_current_pgrp_orphaned(void)
{
    int retval;

    read_lock(&tasklist_lock);
-    retval = will_become_orphaned_pgrp(find_pid(pgrp), NULL);
+    retval = will_become_orphaned_pgrp(task_pgrp(current), NULL);
    read_unlock(&tasklist_lock);

    return retval;
diff --git a/kernel/signal.c b/kernel/signal.c
index 1e34d32..91caafa 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1908,7 +1908,7 @@ relock:

    /* signals can be posted during this window */

-    if (is_orphaned_pgrp(process_group(current)))
+    if (is_current_pgrp_orphaned())
        goto relock;

```

```
spin_lock_irq(&current->sigband->siglock);
```

```
--
```

```
1.4.4.1.g278f
```

---

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>

---