Subject: [PATCH 6/6] c/r: use pids objects for the pgrp/old_pgdp of ttys
Posted by Oren Laadan on Mon, 07 Feb 2011 17:18:08 GMT
View Forum Message <> Reply to Message

Make tty_old_pgrp and tty_pgrp use the shared pid instead of saving
the actual pid number in {checkpoint,restore}_signal().

Signed-off-by: Oren Laadan <orenl@cs.columbia.edu>
---
 kernel/signal.c | 42 ++++++++++++++++++++++++++++++++++++++----------
 1 files changed, 32 insertions(+), 10 deletions(-)

```
diff --git a/kernel/signal.c b/kernel/signal.c
index 718b940..5c174a9 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -3218,25 +3218,34 @@ static int checkpoint_signal(struct ckpt_ctx *ctx, struct task_struct *t)

  /* tty */
  if (signal->leader) {
-  h->tty_old_pgrp = ckpt_pid_vnr(ctx, signal->tty_old_pgrp);
+  h->tty_old_pgrp = ckpt_lookup_pid(ctx, signal->tty_old_pgrp);
    tty = tty_kref_get(signal->tty);
    if (tty) {
     /* irq is already disabled */
     spin_lock(&tty->ctrl_lock);
-    h->tty_pgrp = ckpt_pid_vnr(ctx, tty->pgrp);
+    h->tty_pgrp = ckpt_lookup_pid(ctx, tty->pgrp);
     spin_unlock(&tty->ctrl_lock);
-    tty_kref_put(tty);
    }
   }

  unlock_task_sighand(t, &flags);

  /*
+  * tty_pgrp must be in our namespace, since we are the
+  * session leader and could not have unshared pidns !
+  */
+ if (tty && h->tty_pgrp == 0) {
+  ckpt_err(ctx, -EBUSY, "%(T)tty_pgrp outside namespace\n");
+  ret = -EBUSY;
+  goto out;
+ }
+
+ /*
    * If the session is in an ancestor namespace, skip this tty
    * and set tty_objref = 0. It will not be explicitly restored,
```

```
   * but rather inherited from parent pid-ns at restart time.
   */
- if (tty && ckpt_pid_vnr(ctx, tty->session) > 0) {
+ if (tty && ckpt_lookup_pid(ctx, tty->session) > 0) {
   h->tty_objref = checkpoint_obj(ctx, tty, CKPT_OBJ_TTY);
   if (h->tty_objref < 0)
    ret = h->tty_objref;
@@ -3255,6 +3264,7 @@ static int checkpoint_signal(struct ckpt_ctx *ctx, struct task_struct *t)
  list_splice(&shared_pending.list, &signal->shared_pending.list);
  unlock_task_sighand(t, &flags);
  out:
+ tty_kref_put(tty);
  ckpt_hdr_put(ctx, h);
  return ret;
 }
@@ -3339,6 +3349,10 @@ static int restore_signal(struct ckpt_ctx *ctx)

  /* tty - session */
  if (h->tty_objref) {
+  /*
+   * should only get here if we are session leader, but
+   * we don't explicitly check: forced by calls below
+   */
   tty = ckpt_obj_fetch(ctx, h->tty_objref, CKPT_OBJ_TTY);
   if (IS_ERR(tty)) {
    ret = PTR_ERR(tty);
@@ -3354,6 +3368,11 @@ static int restore_signal(struct ckpt_ctx *ctx)
     * If tty_pgrp == CKPT_PID_NULL, below will
     * fail, so no need for explicit test
     */
+   pgrp = ckpt_obj_fetch(ctx, h->tty_pgrp, CKPT_OBJ_PID);
+   if (IS_ERR(pgrp)) {
+    ret = PTR_ERR(pgrp);
+    goto out;
+   }
    ret = do_tiocspgrp(tty, tty_pair_get_tty(tty),
        pid_vnr(pgrp));
    if (ret < 0)
@@ -3382,13 +3401,16 @@ static int restore_signal(struct ckpt_ctx *ctx)
  do_setitimer(ITIMER_VIRTUAL, &itimer, NULL);
  do_setitimer(ITIMER_PROF, &itimer, NULL);

- /* tty - tty_old_pgrp */
- if (current->signal->leader && h->tty_old_pgrp != CKPT_PID_NULL) {
-  rcu_read_lock();
-  pgrp = NULL;  /* temp until next patch */
-  rcu_read_unlock();
-  if (!pgrp)
```

```
+ /*
+  * tty - tty_old_pgrp: only for session leaders and for valid
+  * tty_old_pgrp, ie, within our namespace, and not CKPT_PID_NULL.
+  */
+ if (current->signal->leader && h->tty_old_pgrp > 0) {
+  pgrp = ckpt_obj_fetch(ctx, h->tty_old_pgrp, CKPT_OBJ_PID);
+  if (IS_ERR(pgrp)) {
+   ret = PTR_ERR(pgrp);
    goto out;
+  }
  }

  spin_lock_irq(&current->sighand->siglock);
--
1.7.1
```