Subject: [PATCH 9/12] tty: Update the tty layer to work with struct pid.
Posted by ebiederm on Wed, 13 Dec 2006 11:07:53 GMT
View Forum Message <> Reply to Message

Of kernel subsystems that work with pids the tty layer
is probably the largest consumer. But it has the nice
virtue that the assiation with a session only lasts until
the session leader exits. Which means that no reference
counting is required. So using struct pid winds up
being a simple optimization to avoid hash table lookups.

In the long term the use of pid_nr also ensures that
when we have multiple pid spaces mixed everything will
work correctly.

Signed-off-by: Eric W. Biederman <eric@maxwell.lnxi.com>

```
---
 arch/um/drivers/line.c     |    2 +-
 drivers/char/ip2/ip2main.c |    4 +-
 drivers/char/n_tty.c       |   12 ++--
 drivers/char/tty_io.c      |  129 +++++++++++++++++++++++++-----------------
 drivers/char/vt.c          |    4 +-
 fs/proc/array.c            |    2 +-
 include/linux/init_task.h  |    2 +-
 include/linux/sched.h      |    2 +-
 include/linux/tty.h        |    4 +-
 kernel/fork.c              |    2 +-
 kernel/sys.c               |    1 -
 11 files changed, 95 insertions(+), 69 deletions(-)

diff --git a/arch/um/drivers/line.c b/arch/um/drivers/line.c
index 83301e1..3c1a1d4 100644
--- a/arch/um/drivers/line.c
+++ b/arch/um/drivers/line.c
@@ -737,7 +737,7 @@ static irqreturn_t winch_interrupt(int irq, void *data)
   line = tty->driver_data;
   chan_window_size(&line->chan_list, &tty->winsize.ws_row,
      &tty->winsize.ws_col);
-  kill_pg(tty->pgrp, SIGWINCH, 1);
+  kill_pgrp(tty->pgrp, SIGWINCH, 1);
  }
  out:
  if(winch->fd != -1)
diff --git a/drivers/char/ip2/ip2main.c b/drivers/char/ip2/ip2main.c
index 7c70310..83c7258 100644
--- a/drivers/char/ip2/ip2main.c
+++ b/drivers/char/ip2/ip2main.c
@@ -1271,8 +1271,8 @@ static void do_input(struct work_struct *work)
```

```
 // code duplicated from n_tty (ldisc)
 static inline void  isig(int sig, struct tty_struct *tty, int flush)
 {
- if (tty->pgrp > 0)
-  kill_pg(tty->pgrp, sig, 1);
+ if (tty->pgrp)
+  kill_pgrp(tty->pgrp, sig, 1);
  if (flush || !L_NOFLSH(tty)) {
   if ( tty->ldisc.flush_buffer )
    tty->ldisc.flush_buffer(tty);
diff --git a/drivers/char/n_tty.c b/drivers/char/n_tty.c
index 7f1ded8..2b50a50 100644
--- a/drivers/char/n_tty.c
+++ b/drivers/char/n_tty.c
@@ -579,8 +579,8 @@ static void eraser(unsigned char c, struct tty_struct *tty)

 static inline void isig(int sig, struct tty_struct *tty, int flush)
 {
- if (tty->pgrp > 0)
-  kill_pg(tty->pgrp, sig, 1);
+ if (tty->pgrp)
+  kill_pgrp(tty->pgrp, sig, 1);
  if (flush || !L_NOFLSH(tty)) {
   n_tty_flush_buffer(tty);
   if (tty->driver->flush_buffer)
@@ -1185,13 +1185,13 @@ static int job_control(struct tty_struct *tty, struct file *file)
  /* don't stop on /dev/console */
  if (file->f_op->write != redirected_tty_write &&
      current->signal->tty == tty) {
-  if (tty->pgrp <= 0)
-   printk("read_chan: tty->pgrp <= 0!\n");
-  else if (process_group(current) != tty->pgrp) {
+  if (!tty->pgrp)
+   printk("read_chan: no tty->pgrp!\n");
+  else if (task_pgrp(current) != tty->pgrp) {
    if (is_ignored(SIGTTIN) ||
        is_current_pgrp_orphaned())
     return -EIO;
-   kill_pg(process_group(current), SIGTTIN, 1);
+   kill_pgrp(task_pgrp(current), SIGTTIN, 1);
    return -ERESTARTSYS;
   }
  }
diff --git a/drivers/char/tty_io.c b/drivers/char/tty_io.c
index d8fdf45..fe98f8c 100644
--- a/drivers/char/tty_io.c
+++ b/drivers/char/tty_io.c
@@ -155,7 +155,7 @@ int tty_ioctl(struct inode * inode, struct file * file,
```

```
        unsigned int cmd, unsigned long arg);
 static int tty_fasync(int fd, struct file * filp, int on);
 static void release_mem(struct tty_struct *tty, int idx);
-static void __proc_set_tty(struct task_struct *tsk, struct tty_struct *tty);
+static struct pid *__proc_set_tty(struct task_struct *tsk, struct tty_struct *tty);

 /**
  * alloc_tty_struct - allocate a tty object
@@ -1110,17 +1110,17 @@ int tty_check_change(struct tty_struct * tty)
 {
  if (current->signal->tty != tty)
   return 0;
- if (tty->pgrp <= 0) {
-  printk(KERN_WARNING "tty_check_change: tty->pgrp <= 0!\n");
+ if (!tty->pgrp) {
+  printk(KERN_WARNING "tty_check_change: tty->pgrp == NULL!\n");
   return 0;
  }
- if (process_group(current) == tty->pgrp)
+ if (task_pgrp(current) == tty->pgrp)
   return 0;
  if (is_ignored(SIGTTOU))
   return 0;
  if (is_current_pgrp_orphaned())
   return -EIO;
- (void) kill_pg(process_group(current), SIGTTOU, 1);
+ (void) kill_pgrp(task_pgrp(current), SIGTTOU, 1);
  return -ERESTARTSYS;
 }

@@ -1355,8 +1355,8 @@ static void do_tty_hangup(struct work_struct *work)
    tty_release is called */

  read_lock(&tasklist_lock);
- if (tty->session > 0) {
-  do_each_task_pid(tty->session, PIDTYPE_SID, p) {
+ if (tty->session) {
+  do_each_pid_task(tty->session, PIDTYPE_SID, p) {
    spin_lock_irq(&p->sighand->siglock);
    if (p->signal->tty == tty)
     p->signal->tty = NULL;
@@ -1366,16 +1366,17 @@ static void do_tty_hangup(struct work_struct *work)
    }
    __group_send_sig_info(SIGHUP, SEND_SIG_PRIV, p);
    __group_send_sig_info(SIGCONT, SEND_SIG_PRIV, p);
-   if (tty->pgrp > 0)
-    p->signal->tty_old_pgrp = tty->pgrp;
+   put_pid(p->signal->tty_old_pgrp);  /* A noop */
```

```
+   if (tty->pgrp)
+    p->signal->tty_old_pgrp = get_pid(tty->pgrp);
    spin_unlock_irq(&p->sighand->siglock);
-  } while_each_task_pid(tty->session, PIDTYPE_SID, p);
+  } while_each_pid_task(tty->session, PIDTYPE_SID, p);
  }
  read_unlock(&tasklist_lock);

  tty->flags = 0;
- tty->session = 0;
- tty->pgrp = -1;
+ tty->session = NULL;
+ tty->pgrp = NULL;
  tty->ctrl_status = 0;
  /*
   * If one of the devices matches a console pointer, we
@@ -1460,12 +1461,12 @@ int tty_hung_up_p(struct file * filp)

 EXPORT_SYMBOL(tty_hung_up_p);

-static void session_clear_tty(pid_t session)
+static void session_clear_tty(struct pid *session)
 {
  struct task_struct *p;
- do_each_task_pid(session, PIDTYPE_SID, p) {
+ do_each_pid_task(session, PIDTYPE_SID, p) {
   proc_clear_tty(p);
- } while_each_task_pid(session, PIDTYPE_SID, p);
+ } while_each_pid_task(session, PIDTYPE_SID, p);
 }

 /**
@@ -1495,48 +1496,54 @@ static void session_clear_tty(pid_t session)
 void disassociate_ctty(int on_exit)
 {
  struct tty_struct *tty;
- int tty_pgrp = -1;
+ struct pid *tty_pgrp = NULL;

  lock_kernel();

  mutex_lock(&tty_mutex);
  tty = get_current_tty();
  if (tty) {
-  tty_pgrp = tty->pgrp;
+  tty_pgrp = get_pid(tty->pgrp);
   mutex_unlock(&tty_mutex);
   /* XXX: here we race, there is nothing protecting tty */
```

```
   if (on_exit && tty->driver->type != TTY_DRIVER_TYPE_PTY)
    tty_vhangup(tty);
  } else if (on_exit) {
-  pid_t old_pgrp;
+  struct pid *old_pgrp;
   spin_lock_irq(&current->sighand->siglock);
   old_pgrp = current->signal->tty_old_pgrp;
-  current->signal->tty_old_pgrp = 0;
+  current->signal->tty_old_pgrp = NULL;
   spin_unlock_irq(&current->sighand->siglock);
   if (old_pgrp) {
-   kill_pg(old_pgrp, SIGHUP, on_exit);
-   kill_pg(old_pgrp, SIGCONT, on_exit);
+   kill_pgrp(old_pgrp, SIGHUP, on_exit);
+   kill_pgrp(old_pgrp, SIGCONT, on_exit);
+   put_pid(old_pgrp);
   }
   mutex_unlock(&tty_mutex);
   unlock_kernel();
   return;
  }
- if (tty_pgrp > 0) {
-  kill_pg(tty_pgrp, SIGHUP, on_exit);
+ if (tty_pgrp) {
+  kill_pgrp(tty_pgrp, SIGHUP, on_exit);
   if (!on_exit)
-   kill_pg(tty_pgrp, SIGCONT, on_exit);
+   kill_pgrp(tty_pgrp, SIGCONT, on_exit);
+  put_pid(tty_pgrp);
  }

  spin_lock_irq(&current->sighand->siglock);
+ tty_pgrp = current->signal->tty_old_pgrp;
  current->signal->tty_old_pgrp = 0;
  spin_unlock_irq(&current->sighand->siglock);
+ put_pid(tty_pgrp);

  mutex_lock(&tty_mutex);
  /* It is possible that do_tty_hangup has free'd this tty */
  tty = get_current_tty();
  if (tty) {
-  tty->session = 0;
-  tty->pgrp = 0;
+  put_pid(tty->session);
+  put_pid(tty->pgrp);
+  tty->session = NULL;
+  tty->pgrp = NULL;
  } else {
```

```
 #ifdef TTY_DEBUG_HANGUP
   printk(KERN_DEBUG "error attempted to write to tty [0x%p]"
@@ -1547,7 +1554,7 @@ void disassociate_ctty(int on_exit)

  /* Now clear signal->tty under the lock */
  read_lock(&tasklist_lock);
- session_clear_tty(process_session(current));
+ session_clear_tty(task_session(current));
  read_unlock(&tasklist_lock);
  unlock_kernel();
 }
@@ -2484,6 +2491,7 @@ static int tty_open(struct inode * inode, struct file * filp)
  int index;
  dev_t device = inode->i_rdev;
  unsigned short saved_flags = filp->f_flags;
+ struct pid *old_pgrp;

  nonseekable_open(inode, filp);

@@ -2577,15 +2585,17 @@ got_driver:
   goto retry_open;
  }

+ old_pgrp = NULL;
  mutex_lock(&tty_mutex);
  spin_lock_irq(&current->sighand->siglock);
  if (!noctty &&
      current->signal->leader &&
      !current->signal->tty &&
-     tty->session == 0)
-  __proc_set_tty(current, tty);
+     tty->session == NULL)
+  old_pgrp = __proc_set_tty(current, tty);
  spin_unlock_irq(&current->sighand->siglock);
  mutex_unlock(&tty_mutex);
+ put_pid(old_pgrp);
  return 0;
 }

@@ -2724,9 +2734,18 @@ static int tty_fasync(int fd, struct file * filp, int on)
   return retval;

  if (on) {
+  enum pid_type type;
+  struct pid *pid;
   if (!waitqueue_active(&tty->read_wait))
    tty->minimum_to_wake = 1;
-  retval = f_setown(filp, (-tty->pgrp) ? : current->pid, 0);
```

```
+  if (tty->pgrp) {
+   pid = tty->pgrp;
+   type = PIDTYPE_PGID;
+  } else {
+   pid = task_pid(current);
+   type = PIDTYPE_PID;
+  }
+  retval = __f_setown(filp, pid, type, 0);
   if (retval)
    return retval;
  } else {
@@ -2828,10 +2847,10 @@ static int tiocswinsz(struct tty_struct *tty, struct tty_struct *real_tty,
   }
  }
 #endif
- if (tty->pgrp > 0)
-  kill_pg(tty->pgrp, SIGWINCH, 1);
- if ((real_tty->pgrp != tty->pgrp) && (real_tty->pgrp > 0))
-  kill_pg(real_tty->pgrp, SIGWINCH, 1);
+ if (tty->pgrp)
+  kill_pgrp(tty->pgrp, SIGWINCH, 1);
+ if ((real_tty->pgrp != tty->pgrp) && real_tty->pgrp)
+  kill_pgrp(real_tty->pgrp, SIGWINCH, 1);
  tty->winsize = tmp_ws;
  real_tty->winsize = tmp_ws;
 done:
@@ -2916,8 +2935,7 @@ static int fionbio(struct file *file, int __user *p)
 static int tiocsctty(struct tty_struct *tty, int arg)
 {
  int ret = 0;
- if (current->signal->leader &&
-  (process_session(current) == tty->session))
+ if (current->signal->leader && (task_session(current) == tty->session))
   return ret;

  mutex_lock(&tty_mutex);
@@ -2930,7 +2948,7 @@ static int tiocsctty(struct tty_struct *tty, int arg)
   goto unlock;
  }

- if (tty->session > 0) {
+ if (tty->session) {
   /*
    * This tty is already the controlling
    * tty for another session group!
@@ -2973,7 +2991,7 @@ static int tiocgpgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t
   */
  if (tty == real_tty && current->signal->tty != real_tty)
```

```
   return -ENOTTY;
- return put_user(real_tty->pgrp, p);
+ return put_user(pid_nr(real_tty->pgrp), p);
 }

 /**
@@ -3000,7 +3018,7 @@ static int tiocspgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t
   return retval;
  if (!current->signal->tty ||
      (current->signal->tty != real_tty) ||
-     (real_tty->session != process_session(current)))
+     (real_tty->session != task_session(current)))
   return -ENOTTY;
  if (get_user(pgrp_nr, p))
   return -EFAULT;
@@ -3015,7 +3033,8 @@ static int tiocspgrp(struct tty_struct *tty, struct tty_struct *real_tty, pid_t
  if (session_of_pgrp(pgrp) != task_session(current))
   goto out_unlock;
  retval = 0;
- real_tty->pgrp = pgrp_nr;
+ put_pid(real_tty->pgrp);
+ real_tty->pgrp = get_pid(pgrp);
 out_unlock:
  rcu_read_unlock();
  return retval;
@@ -3041,9 +3060,9 @@ static int tiocgsid(struct tty_struct *tty, struct tty_struct *real_tty, pid_t _
  */
  if (tty == real_tty && current->signal->tty != real_tty)
   return -ENOTTY;
- if (real_tty->session <= 0)
+ if (!real_tty->session)
   return -ENOTTY;
- return put_user(real_tty->session, p);
+ return put_user(pid_nr(real_tty->session), p);
 }

 /**
@@ -3343,7 +3362,7 @@ void __do_SAK(struct tty_struct *tty)
  tty_hangup(tty);
 #else
  struct task_struct *g, *p;
- int session;
+ struct pid *session;
  int  i;
  struct file *filp;
  struct tty_ldisc *disc;
@@ -3364,12 +3383,12 @@ void __do_SAK(struct tty_struct *tty)
```

```
  read_lock(&tasklist_lock);
  /* Kill the entire session */
- do_each_task_pid(session, PIDTYPE_SID, p) {
+ do_each_pid_task(session, PIDTYPE_SID, p) {
    printk(KERN_NOTICE "SAK: killed process %d"
     " (%s): process_session(p)==tty->session\n",
     p->pid, p->comm);
    send_sig(SIGKILL, p, 1);
- } while_each_task_pid(session, PIDTYPE_SID, p);
+ } while_each_pid_task(session, PIDTYPE_SID, p);
  /* Now kill any processes that happen to have the
   * tty open.
   */
@@ -3538,7 +3557,8 @@ static void initialize_tty_struct(struct tty_struct *tty)
  memset(tty, 0, sizeof(struct tty_struct));
  tty->magic = TTY_MAGIC;
  tty_ldisc_assign(tty, tty_ldisc_get(N_TTY));
- tty->pgrp = -1;
+ tty->session = NULL;
+ tty->pgrp = NULL;
  tty->overrun_time = jiffies;
  tty->buf.head = tty->buf.tail = NULL;
  tty_buffer_init(tty);
@@ -3809,21 +3829,28 @@ void proc_clear_tty(struct task_struct *p)
 }
 EXPORT_SYMBOL(proc_clear_tty);

-static void __proc_set_tty(struct task_struct *tsk, struct tty_struct *tty)
+static struct pid *__proc_set_tty(struct task_struct *tsk, struct tty_struct *tty)
 {
+ struct pid *old_pgrp;
  if (tty) {
-  tty->session = process_session(tsk);
-  tty->pgrp = process_group(tsk);
+  tty->session = get_pid(task_session(tsk));
+  tty->pgrp = get_pid(task_pgrp(tsk));
  }
+ old_pgrp = tsk->signal->tty_old_pgrp;
  tsk->signal->tty = tty;
- tsk->signal->tty_old_pgrp = 0;
+ tsk->signal->tty_old_pgrp = NULL;
+ return old_pgrp;
 }

 void proc_set_tty(struct task_struct *tsk, struct tty_struct *tty)
 {
+ struct pid *old_pgrp;
+
```

```
  spin_lock_irq(&tsk->sighand->siglock);
- __proc_set_tty(tsk, tty);
+ old_pgrp = __proc_set_tty(tsk, tty);
  spin_unlock_irq(&tsk->sighand->siglock);
+
+ put_pid(old_pgrp);
 }

 struct tty_struct *get_current_tty(void)
diff --git a/drivers/char/vt.c b/drivers/char/vt.c
index a8239da..4d52cab 100644
--- a/drivers/char/vt.c
+++ b/drivers/char/vt.c
@@ -869,8 +869,8 @@ int vc_resize(struct vc_data *vc, unsigned int cols, unsigned int lines)
   ws.ws_col = vc->vc_cols;
   ws.ws_ypixel = vc->vc_scan_lines;
   if ((ws.ws_row != cws->ws_row || ws.ws_col != cws->ws_col) &&
-      vc->vc_tty->pgrp > 0)
-   kill_pg(vc->vc_tty->pgrp, SIGWINCH, 1);
+      vc->vc_tty->pgrp)
+   kill_pgrp(vc->vc_tty->pgrp, SIGWINCH, 1);
   *cws = ws;
  }

diff --git a/fs/proc/array.c b/fs/proc/array.c
index 70e4fab..07c9cdb 100644
--- a/fs/proc/array.c
+++ b/fs/proc/array.c
@@ -351,7 +351,7 @@ static int do_task_stat(struct task_struct *task, char * buffer, int whole)
   struct signal_struct *sig = task->signal;

   if (sig->tty) {
-   tty_pgrp = sig->tty->pgrp;
+   tty_pgrp = pid_nr(sig->tty->pgrp);
    tty_nr = new_encode_dev(tty_devnum(sig->tty));
   }

diff --git a/include/linux/init_task.h b/include/linux/init_task.h
index b531515..9991297 100644
--- a/include/linux/init_task.h
+++ b/include/linux/init_task.h
@@ -66,7 +66,7 @@
  .cpu_timers = INIT_CPU_TIMERS(sig.cpu_timers),  \
  .rlim  = INIT_RLIMITS,     \
  .pgrp  = 1,      \
- .tty_old_pgrp   = 0,      \
+ .tty_old_pgrp   = NULL,      \
  { .__session      = 1},      \
```

```
 }

diff --git a/include/linux/sched.h b/include/linux/sched.h
index ea92e5c..ef0d6fe 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -436,7 +436,7 @@ struct signal_struct {

  /* job control IDs */
  pid_t pgrp;
- pid_t tty_old_pgrp;
+ struct pid *tty_old_pgrp;

  union {
   pid_t session __deprecated;
diff --git a/include/linux/tty.h b/include/linux/tty.h
index 13a4918..f07e390 100644
--- a/include/linux/tty.h
+++ b/include/linux/tty.h
@@ -177,8 +177,8 @@ struct tty_struct {
  struct mutex termios_mutex;
  struct ktermios *termios, *termios_locked;
  char name[64];
- int pgrp;
- int session;
+ struct pid *pgrp;
+ struct pid *session;
  unsigned long flags;
  int count;
  struct winsize winsize;
diff --git a/kernel/fork.c b/kernel/fork.c
index d16c566..24b4af0 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -869,7 +869,7 @@ static inline int copy_signal(unsigned long clone_flags, struct task_struct
* ts
  sig->it_prof_incr = cputime_zero;

  sig->leader = 0; /* session leadership doesn't inherit */
- sig->tty_old_pgrp = 0;
+ sig->tty_old_pgrp = NULL;

  sig->utime = sig->stime = sig->cutime = sig->cstime = cputime_zero;
  sig->nvcsw = sig->nivcsw = sig->cnvcsw = sig->cnivcsw = 0;
diff --git a/kernel/sys.c b/kernel/sys.c
index c7675c1..a8f2ebe 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
```

```
@@ -1503,7 +1503,6 @@ asmlinkage long sys_setsid(void)

  spin_lock(&group_leader->sighand->siglock);
  group_leader->signal->tty = NULL;
- group_leader->signal->tty_old_pgrp = 0;
  spin_unlock(&group_leader->sighand->siglock);

  err = process_group(group_leader);
--
1.4.4.1.g278f
```