Subject: [PATCH] usb: Fixup usb so it uses struct pid
Posted by ebiederm on Sun, 10 Sep 2006 04:42:10 GMT
View Forum Message <> Reply to Message

The problem by remember a user space process by it's pid it is
possible that the process will exit, pid wrap around will occur and a
different process will appear in it's place. Holding a reference
to a struct pid avoid that problem, and paves the way
for implementing a pid namespace.

Also since usb is the only user of kill_proc_info_as_uid
rename kill_proc_info_as_uid to kill_pid_info_as_uid
and have the new version take a struct pid.

This patch is against 2.6.18-rc6-mm1.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 drivers/usb/core/devio.c | 10 ++++++----
 drivers/usb/core/inode.c | 2 +-
 drivers/usb/core/usb.h | 2 +-
 include/linux/sched.h | 2 +-
 kernel/signal.c | 8 ++++----
 5 files changed, 13 insertions(+), 11 deletions(-)

diff --git a/drivers/usb/core/devio.c b/drivers/usb/core/devio.c
index 3ef2778..93087c7 100644
--- a/drivers/usb/core/devio.c
+++ b/drivers/usb/core/devio.c
@@ -65,7 +65,7 @@ DEFINE_MUTEX(usbfs_mutex);
 struct async {
  struct list_head asynclist;
  struct dev_state *ps;
- pid_t pid;
+ struct pid *pid;
  uid_t uid, euid;
  unsigned int signr;
  unsigned int ifnum;
@@ -225,6 +225,7 @@ static struct async *alloc_async(unsigne

 static void free_async(struct async *as)
 {
+ put_pid(as->pid);
  kfree(as->urb->transfer_buffer);
  kfree(as->urb->setup_packet);
  usb_free_urb(as->urb);
@@ -317,7 +318,7 @@ static void async_completed(struct urb *
   sinfo.si_errno = as->urb->status;

```
   sinfo.si_code = SI_ASYNCIO;
   sinfo.si_addr = as->userurb;
-  kill_proc_info_as_uid(as->signr, &sinfo, as->pid, as->uid,
+  kill_pid_info_as_uid(as->signr, &sinfo, as->pid, as->uid,
          as->euid, as->secid);
  }
  snoop(&urb->dev->dev, "urb complete\n");
@@ -572,7 +573,7 @@ static int usbdev_open(struct inode *ino
  INIT_LIST_HEAD(&ps->async_completed);
  init_waitqueue_head(&ps->wait);
  ps->discsignr = 0;
- ps->disc_pid = current->pid;
+ ps->disc_pid = get_pid(task_pid(current));
  ps->disc_uid = current->uid;
  ps->disc_euid = current->euid;
  ps->disccontext = NULL;
@@ -610,6 +611,7 @@ static int usbdev_release(struct inode *
  usb_autosuspend_device(dev, 1);
  usb_unlock_device(dev);
  usb_put_dev(dev);
+ put_pid(ps->disc_pid);
  kfree(ps);
  return 0;
 }
@@ -1062,7 +1064,7 @@ static int proc_do_submiturb(struct dev_
   as->userbuffer = NULL;
  as->signr = uurb->signr;
  as->ifnum = ifnum;
- as->pid = current->pid;
+ as->pid = get_pid(task_pid(current));
  as->uid = current->uid;
  as->euid = current->euid;
  security_task_getsecid(current, &as->secid);
diff --git a/drivers/usb/core/inode.c b/drivers/usb/core/inode.c
index 7c77c2d..b5d6a79 100644
--- a/drivers/usb/core/inode.c
+++ b/drivers/usb/core/inode.c
@@ -699,7 +699,7 @@ static void usbfs_remove_device(struct u
    sinfo.si_errno = EPIPE;
    sinfo.si_code = SI_ASYNCIO;
    sinfo.si_addr = ds->disccontext;
-   kill_proc_info_as_uid(ds->discsignr, &sinfo, ds->disc_pid, ds->disc_uid, ds->disc_euid,
ds->secid);
+   kill_pid_info_as_uid(ds->discsignr, &sinfo, ds->disc_pid, ds->disc_uid, ds->disc_euid,
ds->secid);
   }
  }
 }
```

```
diff --git a/drivers/usb/core/usb.h b/drivers/usb/core/usb.h
index e8bc2e4..e324164 100644
--- a/drivers/usb/core/usb.h
+++ b/drivers/usb/core/usb.h
@@ -122,7 +122,7 @@ struct dev_state {
  struct list_head async_completed;
  wait_queue_head_t wait;     /* wake up if a request completed */
  unsigned int discsignr;
- pid_t disc_pid;
+ struct pid *disc_pid;
  uid_t disc_uid, disc_euid;
  void __user *disccontext;
  unsigned long ifclaimed;
diff --git a/include/linux/sched.h b/include/linux/sched.h
index f11b874..36031ef 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1303,12 +1303,12 @@ extern int force_sig_info(int, struct si
 extern int __kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
 extern int kill_pgrp_info(int sig, struct siginfo *info, struct pid *pgrp);
 extern int kill_pid_info(int sig, struct siginfo *info, struct pid *pid);
+extern int kill_pid_info_as_uid(int, struct siginfo *, struct pid *, uid_t, uid_t, u32);
 extern int kill_pgrp(struct pid *pid, int sig, int priv);
 extern int kill_pid(struct pid *pid, int sig, int priv);
 extern int __kill_pg_info(int sig, struct siginfo *info, pid_t pgrp);
 extern int kill_pg_info(int, struct siginfo *, pid_t);
 extern int kill_proc_info(int, struct siginfo *, pid_t);
-extern int kill_proc_info_as_uid(int, struct siginfo *, pid_t, uid_t, uid_t, u32);
 extern void do_notify_parent(struct task_struct *, int);
 extern void force_sig(int, struct task_struct *);
 extern void force_sig_specific(int, struct task_struct *);
diff --git a/kernel/signal.c b/kernel/signal.c
index e04fd9e..0175b24 100644
--- a/kernel/signal.c
+++ b/kernel/signal.c
@@ -1260,8 +1260,8 @@ kill_proc_info(int sig, struct siginfo *
  return error;
 }

-/* like kill_proc_info(), but doesn't use uid/euid of "current" */
-int kill_proc_info_as_uid(int sig, struct siginfo *info, pid_t pid,
+/* like kill_pid_info(), but doesn't use uid/euid of "current" */
+int kill_pid_info_as_uid(int sig, struct siginfo *info, struct pid *pid,
         uid_t uid, uid_t euid, u32 secid)
 {
  int ret = -EINVAL;
@@ -1271,7 +1271,7 @@ int kill_proc_info_as_uid(int sig, struc
   return ret;
```

```
  read_lock(&tasklist_lock);
- p = find_task_by_pid(pid);
+ p = pid_task(pid, PIDTYPE_PID);
  if (!p) {
   ret = -ESRCH;
   goto out_unlock;
@@ -1295,7 +1295,7 @@ out_unlock:
  read_unlock(&tasklist_lock);
  return ret;
 }
-EXPORT_SYMBOL_GPL(kill_proc_info_as_uid);
+EXPORT_SYMBOL_GPL(kill_pid_info_as_uid);

 /*
  * kill_something_info() interprets pid in interesting ways just like kill(2).
--
1.4.2.rc3.g7e18e-dirty
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers