Subject: [PATCH] Use find_task_by_pid_ns() in places that operate with virtual pids
Posted by Pavel Emelianov on Fri, 17 Aug 2007 09:34:29 GMT
View Forum Message <> Reply to Message

When the pid comes from the userspace, the find_task_by_pid_ns()
should be used to find the task by pid in particular (usually the
current) namespace. These places were lost in earlier patches.

Think over: all these places work like this:

```
 if (pid == 0)
  task = current;
 else
  task = find_task_by_pid_ns(pid);
```

the question is: does it worth introducing a common helper for such
case and (if it does) what should its name be?

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>
Cc: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Cc: Oleg Nesterov <oleg@tv-sign.ru>

This is a lost hunks of the -mm patch named
pid-namespaces-changes-to-show-virtual-ids-to-user.patch

```
 fs/ioprio.c          |    6 ++++--
 kernel/futex.c       |    6 ++++--
 kernel/futex_compat.c |    3 ++-
 kernel/sched.c       |    3 ++-
 mm/mempolicy.c       |    3 ++-
 mm/migrate.c         |    3 ++-
 6 files changed, 16 insertions(+), 8 deletions(-)

diff --git a/fs/ioprio.c b/fs/ioprio.c
index c99fe7b..0a615f8 100644
--- a/fs/ioprio.c
+++ b/fs/ioprio.c
@@ -94,7 +94,8 @@ asmlinkage long sys_ioprio_set(int which
    if (!who)
     p = current;
    else
-    p = find_task_by_pid(who);
+    p = find_task_by_pid_ns(who,
+      current->nsproxy->pid_ns);
    if (p)
```

```
     ret = set_task_ioprio(p, ioprio);
    break;
@@ -181,7 +182,8 @@ asmlinkage long sys_ioprio_get(int which
    if (!who)
     p = current;
    else
-    p = find_task_by_pid(who);
+    p = find_task_by_pid_ns(who,
+      current->nsproxy->pid_ns);
    if (p)
     ret = get_task_ioprio(p);
    break;
diff --git a/kernel/futex.c b/kernel/futex.c
index c61634a..0c1b777 100644
--- a/kernel/futex.c
+++ b/kernel/futex.c
@@ -444,7 +444,8 @@ static struct task_struct * futex_find_g
  struct task_struct *p;

  rcu_read_lock();
- p = find_task_by_pid(pid);
+ p = find_task_by_pid_ns(pid,
+   current->nsproxy->pid_ns);

  if (!p || ((current->euid != p->euid) && (current->euid != p->uid)))
   p = ERR_PTR(-ESRCH);
@@ -1855,7 +1856,8 @@ sys_get_robust_list(int pid, struct robu

   ret = -ESRCH;
   rcu_read_lock();
-  p = find_task_by_pid(pid);
+  p = find_task_by_pid_ns(pid,
+    current->nsproxy->pid_ns);
   if (!p)
    goto err_unlock;
   ret = -EPERM;
diff --git a/kernel/futex_compat.c b/kernel/futex_compat.c
index f792136..e7563bf 100644
--- a/kernel/futex_compat.c
+++ b/kernel/futex_compat.c
@@ -116,7 +116,8 @@ compat_sys_get_robust_list(int pid, comp

   ret = -ESRCH;
   read_lock(&tasklist_lock);
-  p = find_task_by_pid(pid);
+  p = find_task_by_pid_ns(pid,
+    current->nsproxy->pid_ns);
   if (!p)
```

```
    goto err_unlock;
   ret = -EPERM;
diff --git a/kernel/sched.c b/kernel/sched.c
index 4359b6b..1f81803 100644
--- a/kernel/sched.c
+++ b/kernel/sched.c
@@ -4108,7 +4108,8 @@ struct task_struct *idle_task(int cpu)
  */
 static inline struct task_struct *find_process_by_pid(pid_t pid)
 {
- return pid ? find_task_by_pid(pid) : current;
+ return pid ?
+  find_task_by_pid_ns(pid, current->nsproxy->pid_ns) : current;
 }

 /* Actually do priority change: must hold rq lock. */
diff --git a/mm/mempolicy.c b/mm/mempolicy.c
index e7eb2bb..934abc9 100644
--- a/mm/mempolicy.c
+++ b/mm/mempolicy.c
@@ -930,7 +930,8 @@ asmlinkage long sys_migrate_pages(pid_t

  /* Find the mm_struct */
  read_lock(&tasklist_lock);
- task = pid ? find_task_by_pid(pid) : current;
+ task = pid ?
+  find_task_by_pid_ns(pid, current->nsproxy->pid_ns) : current;
  if (!task) {
   read_unlock(&tasklist_lock);
   return -ESRCH;
diff --git a/mm/migrate.c b/mm/migrate.c
index 477b894..7f43c5f 100644
--- a/mm/migrate.c
+++ b/mm/migrate.c
@@ -917,7 +917,8 @@ asmlinkage long sys_move_pages(pid_t pid

  /* Find the mm_struct */
  read_lock(&tasklist_lock);
- task = pid ? find_task_by_pid(pid) : current;
+ task = pid ?
+  find_task_by_pid_ns(pid, current->nsproxy->pid_ns) : current;
  if (!task) {
   read_unlock(&tasklist_lock);
   return -ESRCH;
```