Subject: [PATCH 4/28] [PREP 4/14] Make find_ge_pid() operate on virtual pids
Posted by Pavel Emelianov on Fri, 15 Jun 2007 16:02:49 GMT
View Forum Message <> Reply to Message

find_ge_pid() is used in proc readdir and thus must be able to work with
virtual pids as well. The numerical ids are shown depending of what
namespace owns this proc mount. It assumes that the namespace in question
is stored in the superblock's private data.

Proc support will come later in this set, but this patch is logically tied
to the previous ones and git bisect safe, so it goes here.

Signed-off-by: Pavel Emelianov <xemul@openvz.org>

```
 fs/proc/base.c      |   15 +++++++++------
 include/linux/pid.h |    2 +-
 kernel/pid.c        |    6 +++---
 3 files changed, 13 insertions(+), 10 deletions(-)

--- ./fs/proc/base.c.findgepid 2007-06-15 15:00:32.000000000 +0400
+++ ./fs/proc/base.c 2007-06-15 15:00:59.000000000 +0400
@@ -2291,7 +2291,8 @@ out:
  * Find the first task with tgid >= tgid
  *
  */
-static struct task_struct *next_tgid(unsigned int tgid)
+static struct task_struct *next_tgid(unsigned int tgid,
+  struct pid_namespace *ns)
 {
  struct task_struct *task;
  struct pid *pid;
@@ -2299,9 +2300,9 @@ static struct task_struct *next_tgid(uns
  rcu_read_lock();
 retry:
  task = NULL;
- pid = find_ge_pid(tgid);
+ pid = find_ge_pid(tgid, ns);
  if (pid) {
-  tgid = pid->nr + 1;
+  tgid = pid_nr_ns(pid, ns) + 1;
   task = pid_task(pid, PIDTYPE_PID);
   /* What we to know is if the pid we have find is the
    * pid of a thread_group_leader.  Testing for task
@@ -2341,6 +2342,7 @@ int proc_pid_readdir(struct file * filp,
  struct task_struct *reaper = get_proc_task(filp->f_path.dentry->d_inode);
  struct task_struct *task;
```

```
  int tgid;
+ struct pid_namespace *ns;

  if (!reaper)
   goto out_no_task;
@@ -2351,11 +2353,12 @@ int proc_pid_readdir(struct file * filp,
    goto out;
  }

+ ns = (struct pid_namespace *)filp->f_dentry->d_sb->s_fs_info;
  tgid = filp->f_pos - TGID_OFFSET;
- for (task = next_tgid(tgid);
+ for (task = next_tgid(tgid, ns);
       task;
-      put_task_struct(task), task = next_tgid(tgid + 1)) {
-  tgid = task->pid;
+      put_task_struct(task), task = next_tgid(tgid + 1, ns)) {
+  tgid = task_pid_nr_ns(task, ns);
   filp->f_pos = tgid + TGID_OFFSET;
   if (proc_pid_fill_cache(filp, dirent, filldir, task, tgid) < 0) {
    put_task_struct(task);
--- ./include/linux/pid.h.findgepid 2007-06-15 15:00:44.000000000 +0400
+++ ./include/linux/pid.h 2007-06-15 15:00:59.000000000 +0400
@@ -105,7 +105,7 @@ extern struct pid *FASTCALL(find_pid_ns(
  * Lookup a PID in the hash table, and return with it's count elevated.
  */
 extern struct pid *find_get_pid(int nr);
-extern struct pid *find_ge_pid(int nr);
+extern struct pid *find_ge_pid(int nr, struct pid_namespace *);

 extern struct pid *alloc_pid(void);
 extern void FASTCALL(free_pid(struct pid *pid));
--- ./kernel/pid.c.findgepid 2007-06-15 15:00:44.000000000 +0400
+++ ./kernel/pid.c 2007-06-15 15:00:59.000000000 +0400
@@ -351,15 +351,15 @@ struct pid *find_get_pid(pid_t nr)
  *
  * If there is a pid at nr this function is exactly the same as find_pid.
  */
-struct pid *find_ge_pid(int nr)
+struct pid *find_ge_pid(int nr, struct pid_namespace *ns)
 {
  struct pid *pid;

  do {
-  pid = find_pid(nr);
+  pid = find_pid_ns(nr, ns);
   if (pid)
    break;
```

```
-  nr = next_pidmap(current->nsproxy->pid_ns, nr);
+  nr = next_pidmap(ns, nr);
  } while (nr > 0);

  return pid;
```

Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers