Subject: [PATCH 10/20] Helpers to find the task by its numerical ids
Posted by Pavel Emelianov on Tue, 07 Aug 2007 09:29:54 GMT
View Forum Message <> Reply to Message

When searching the task by numerical id on may need to find it using global
pid (as it is done now in kernel) or by its virtual id, e.g. when sending
a signal to a task from one namespace the sender will specify the task's
virtual id and we should find the task by this value.

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

```
 fs/proc/base.c        |    2 +-
 include/linux/pid.h   |   16 +++++++++++++--
 include/linux/sched.h |   31 +++++++++++++++++++++++++++++--
 kernel/pid.c          |   42 ++++++++++++++++++++++++++++++++----------
 4 files changed, 69 insertions(+), 22 deletions(-)

--- ./fs/proc/base.c.ve10 2007-08-06 18:59:11.000000000 +0400
+++ ./fs/proc/base.c 2007-08-06 18:59:11.000000000 +0400
@@ -2244,7 +2244,7 @@ static struct task_struct *next_tgid(uns
  rcu_read_lock();
 retry:
  task = NULL;
- pid = find_ge_pid(tgid);
+ pid = find_ge_pid(tgid, &init_pid_ns);
  if (pid) {
   tgid = pid->nr + 1;
   task = pid_task(pid, PIDTYPE_PID);
--- ./include/linux/pid.h.ve10 2007-08-06 18:59:11.000000000 +0400
+++ ./include/linux/pid.h 2007-08-06 18:59:11.000000000 +0400
@@ -99,17 +99,29 @@ extern void FASTCALL(detach_pid(struct t
 extern void FASTCALL(transfer_pid(struct task_struct *old,
       struct task_struct *new, enum pid_type));

+struct pid_namespace;
+extern struct pid_namespace init_pid_ns;
+
 /*
  * look up a PID in the hash table. Must be called with the tasklist_lock
  * or rcu_read_lock() held.
+ *
+ * find_pid_ns() finds the pid in the namespace specified
+ * find_pid() find the pid by its global id, i.e. in the init namespace
+ * find_vpid() finr the pid by its virtual id, i.e. in the current namespace
+ *
```

```
+ * see also find_task_by_pid() set in include/linux/sched.h
  */
-extern struct pid *FASTCALL(find_pid(int nr));
+extern struct pid *FASTCALL(find_pid_ns(int nr, struct pid_namespace *ns));
+
+#define find_vpid(pid) find_pid_ns(pid, current->nsproxy->pid_ns)
+#define find_pid(pid) find_pid_ns(pid, &init_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(struct pid_namespace *ns);
 extern void FASTCALL(free_pid(struct pid *pid));
--- ./include/linux/sched.h.ve10 2007-08-06 18:59:11.000000000 +0400
+++ ./include/linux/sched.h 2007-08-06 18:59:11.000000000 +0400
@@ -1541,8 +1541,35 @@ extern struct task_struct init_task;

 extern struct    mm_struct init_mm;

-#define find_task_by_pid(nr) find_task_by_pid_type(PIDTYPE_PID, nr)
-extern struct task_struct *find_task_by_pid_type(int type, int pid);
+extern struct pid_namespace init_pid_ns;
+
+/*
+ * find a task by one of its numerical ids
+ *
+ * find_task_by_pid_type_ns():
+ *      it is the most generic call - it finds a task by all id,
+ *      type and namespace specified
+ * find_task_by_pid_ns():
+ *      finds a task by its pid in the specified namespace
+ * find_task_by_pid_type():
+ *      finds a task by its global id with the specified type, e.g.
+ *      by global session id
+ * find_task_by_pid():
+ *      finds a task by its global pid
+ *
+ * see also find_pid() etc in include/linux/pid.h
+ */
+
+extern struct task_struct *find_task_by_pid_type_ns(int type, int pid,
+  struct pid_namespace *ns);
+
+#define find_task_by_pid_ns(nr, ns) \
```

```
+  find_task_by_pid_type_ns(PIDTYPE_PID, nr, ns)
+#define find_task_by_pid_type(type, nr) \
+  find_task_by_pid_type_ns(type, nr, &init_pid_ns)
+#define find_task_by_pid(nr)  \
+  find_task_by_pid_type(PIDTYPE_PID, nr)
+
 extern void __set_special_pids(pid_t session, pid_t pgrp);

 /* per-UID process charging. */
--- ./kernel/pid.c.ve10 2007-08-06 18:59:11.000000000 +0400
+++ ./kernel/pid.c 2007-08-06 18:59:49.000000000 +0400
@@ -212,7 +212,8 @@ fastcall void free_pid(struct pid *pid)
  unsigned long flags;

  spin_lock_irqsave(&pidmap_lock, flags);
- hlist_del_rcu(&pid->pid_chain);
+ for (i = 0; i <= pid->level; i++)
+  hlist_del_rcu(&pid->numbers[i].pid_chain);
  spin_unlock_irqrestore(&pidmap_lock, flags);

  for (i = 0; i <= pid->level; i++)
@@ -227,6 +228,7 @@ struct pid *alloc_pid(struct pid_namespa
  enum pid_type type;
  int i, nr;
  struct pid_namespace *tmp;
+ struct upid *upid;

  pid = kmem_cache_alloc(ns->pid_cachep, GFP_KERNEL);
  if (!pid)
@@ -253,7 +255,11 @@ struct pid *alloc_pid(struct pid_namespa
   INIT_HLIST_HEAD(&pid->tasks[type]);

  spin_lock_irq(&pidmap_lock);
- hlist_add_head_rcu(&pid->pid_chain, &pid_hash[pid_hashfn(pid->nr, ns)]);
+ for (i = ns->level; i >= 0; i--) {
+  upid = &pid->numbers[i];
+  hlist_add_head_rcu(&upid->pid_chain,
+    &pid_hash[pid_hashfn(upid->nr, upid->ns)]);
+ }
  spin_unlock_irq(&pidmap_lock);

 out:
@@ -268,19 +274,20 @@ out_free:
  goto out;
 }

-struct pid * fastcall find_pid(int nr)
+struct pid * fastcall find_pid_ns(int nr, struct pid_namespace *ns)
```

```
 {
  struct hlist_node *elem;
- struct pid *pid;
+ struct upid *pnr;
+
+ hlist_for_each_entry_rcu(pnr, elem,
+   &pid_hash[pid_hashfn(nr, ns)], pid_chain)
+  if (pnr->nr == nr && pnr->ns == ns)
+   return container_of(pnr, struct pid,
+     numbers[ns->level]);

- hlist_for_each_entry_rcu(pid, elem,
-   &pid_hash[pid_hashfn(nr, &init_pid_ns)], pid_chain) {
-  if (pid->nr == nr)
-   return pid;
- }
  return NULL;
 }
-EXPORT_SYMBOL_GPL(find_pid);
+EXPORT_SYMBOL_GPL(find_pid_ns);

 /*
  * attach_pid() must be called with the tasklist_lock write-held.
@@ -340,12 +347,13 @@ struct task_struct * fastcall pid_task(s
 /*
  * Must be called under rcu_read_lock() or with tasklist_lock read-held.
  */
-struct task_struct *find_task_by_pid_type(int type, int nr)
+struct task_struct *find_task_by_pid_type_ns(int type, int nr,
+  struct pid_namespace *ns)
 {
- return pid_task(find_pid(nr), type);
+ return pid_task(find_pid_ns(nr, ns), type);
 }

-EXPORT_SYMBOL(find_task_by_pid_type);
+EXPORT_SYMBOL(find_task_by_pid_type_ns);

 struct pid *get_task_pid(struct task_struct *task, enum pid_type type)
 {
@@ -372,7 +380,7 @@ struct pid *find_get_pid(pid_t nr)
  struct pid *pid;

  rcu_read_lock();
- pid = get_pid(find_pid(nr));
+ pid = get_pid(find_vpid(nr));
  rcu_read_unlock();
```

```
  return pid;
@@ -396,15 +404,15 @@ pid_t pid_nr_ns(struct pid *pid, struct
  *
  * 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(task_active_pid_ns(current), nr);
+  nr = next_pidmap(ns, nr);
  } while (nr > 0);

  return pid;
```