Subject: [PATCH 5/13] Expand the pid/task seeking functions set
Posted by Pavel Emelianov on Thu, 24 May 2007 12:46:49 GMT
View Forum Message <> Reply to Message

We need the extended set of functions for searching
tasks and pids - search in global namespace, in local
namespace (current belongs to) and in arbitrary namespace
(used in proc).

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

```
diff --git a/include/linux/sched.h b/include/linux/sched.h
index d4de6d8..7743a11 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1298,8 +1428,16 @@ 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;
+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. */
diff --git a/include/linux/pid.h b/include/linux/pid.h
index 1e0e4e3..3a30f8a 100644
--- a/include/linux/pid.h
+++ b/include/linux/pid.h
@@ -83,17 +88,20 @@ 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;
 /*
  * look up a PID in the hash table. Must be called with the tasklist_lock
  * or rcu_read_lock() held.
```

```
  */
 extern struct pid *FASTCALL(find_pid(int nr));
+extern struct pid *FASTCALL(__find_vpid(int nr, struct pid_namespace *ns));
+#define find_vpid(pid) __find_vpid(pid, current->nsproxy->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));
diff --git a/kernel/pid.c b/kernel/pid.c
index eb66bd2..1815af4 100644
--- a/kernel/pid.c
+++ b/kernel/pid.c
@@ -249,6 +289,27 @@ struct pid * fastcall find_pid(int nr)
 }
 EXPORT_SYMBOL_GPL(find_pid);

+struct pid * fastcall __find_vpid(int nr, struct pid_namespace *ns)
+{
+#ifdef CONFIG_PID_NS
+ struct hlist_node *elem;
+ struct pid *pid;
+#endif
+
+ if (ns == &init_pid_ns)
+  return find_pid(nr);
+
+#ifdef CONFIG_PID_NS
+ hlist_for_each_entry_rcu(pid, elem,
+   &vpid_hash[vpid_hashfn(nr, ns)], vpid_chain) {
+  if (pid->vnr == nr && pid->ns == ns)
+   return pid;
+ }
+#endif
+ return NULL;
+}
+EXPORT_SYMBOL_GPL(__find_vpid);
+
 /*
  * attach_pid() must be called with the tasklist_lock write-held.
  */
@@ -307,12 +368,13 @@ 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_vpid(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)
 {
@@ -339,7 +401,7 @@ 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;
@@ -350,15 +412,15 @@ 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_vpid(nr, ns);
   if (pid)
    break;
-  nr = next_pidmap(current->nsproxy->pid_ns, nr);
+  nr = next_pidmap(ns, nr);
  } while (nr > 0);

  return pid;
```