Subject: [PATCH 2/28] [PREP 2/14] Helpers to obtain pid numbers
Posted by Pavel Emelianov on Fri, 15 Jun 2007 16:00:23 GMT
View Forum Message <> Reply to Message

When showing pid to user or getting the pid numerical id for in-kernel
use the value of this id may differ depending on the namespace.

This set of helpers is used to get the global pid nr, the virtual (i.e.
seen by task in its namespace) nr and the nr as it is seen from the
specified namespace.

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

```
---

 pid.h   |   26 +++++++++++++
 sched.h |  127 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++------
 2 files changed, 143 insertions(+), 10 deletions(-)

--- ./include/linux/pid.h.nrhelpers 2007-06-15 14:31:19.000000000 +0400
+++ ./include/linux/pid.h 2007-06-15 14:39:27.000000000 +0400
@@ -98,6 +98,20 @@ extern struct pid *find_ge_pid(int nr);
 extern struct pid *alloc_pid(void);
 extern void FASTCALL(free_pid(struct pid *pid));

+struct pid_namespace;
+
+/*
+ * the helpers to get the pid's id seen from different namespaces
+ *
+ * pid_nr()    : global id, i.e. the id seen from the init namespace;
+ * pid_vnr()   : virtual id, i.e. the id seen from the namespace this pid
+ *               belongs to. this only makes sence when called in the
+ *               context of the task that belongs to the same namespace;
+ * pid_nr_ns() : id seen from the ns specified.
+ *
+ * see also task_xid_nr() etc in include/linux/sched.h
+ */
+#ifndef CONFIG_PID_NS
 static inline pid_t pid_nr(struct pid *pid)
 {
  pid_t nr = 0;
@@ -106,6 +120,18 @@ static inline pid_t pid_nr(struct pid *p
  return nr;
 }

+static inline pid_t pid_vnr(struct pid *pid)
+{
```

```
+ return pid_nr(pid);
+}
+
+static inline pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ return pid_nr(pid);
+}
+#else
+#endif
+
 #define do_each_pid_task(pid, type, task)    \
  do {        \
   struct hlist_node *pos___;    \
--- ./include/linux/sched.h.nrhelpers 2007-06-15 14:31:32.000000000 +0400
+++ ./include/linux/sched.h 2007-06-15 14:46:27.000000000 +0400
@@ -1153,16 +1153,6 @@ struct task_struct {
 #endif
 };

-static inline pid_t task_pgrp_nr(struct task_struct *tsk)
-{
- return tsk->signal->pgrp;
-}
-
-static inline pid_t task_session_nr(struct task_struct *tsk)
-{
- return tsk->signal->__session;
-}
-
 static inline void set_task_session(struct task_struct *tsk, pid_t session)
 {
  tsk->signal->__session = session;
@@ -1188,6 +1178,123 @@ static inline struct pid *task_session(s
  return task->group_leader->pids[PIDTYPE_SID].pid;
 }

+struct pid_namespace;
+
+/*
+ * the helpers to get the task's different pids as they are seen
+ * from various namespaces
+ *
+ * task_xid_nr()     : global id, i.e. the id seen from the init namespace;
+ * task_xid_vnr()    : virtual id, i.e. the id seen from the namespace the task
+ *                     belongs to. this only makes sence when called in the
+ *                     context of the task that belongs to the same namespace;
+ * task_xid_nr_ns()  : id seen from the ns specified;
+ *
```

```
+ * set_task_vxid()   : assigns a virtual id to a task;
+ *
+ * task_ppid_nr_ns() : the parent's id as seen from the namespace specified.
+ *                     the result depends on the namespace and whether the
+ *                     task in question is the namespace's init. e.g. for the
+ *                     namespace's init this will return 0 when called from
+ *                     the namespace of this init, or appropriate id otherwise.
+ *
+ *
+ * see also pid_nr() etc in include/linux/pid.h
+ */
+
+#ifndef CONFIG_PID_NS
+static inline pid_t task_pid_nr(struct task_struct *tsk)
+{
+ return tsk->pid;
+}
+
+static inline pid_t task_pid_vnr(struct task_struct *tsk)
+{
+ return task_pid_nr(tsk);
+}
+
+static inline pid_t task_pid_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ return task_pid_nr(tsk);
+}
+
+static inline void set_task_vpid(struct task_struct *tsk, pid_t nr)
+{
+}
+
+
+static inline pid_t task_tgid_nr(struct task_struct *tsk)
+{
+ return tsk->tgid;
+}
+
+static inline pid_t task_tgid_vnr(struct task_struct *tsk)
+{
+ return task_pid_nr(tsk);
+}
+
+static inline pid_t task_tgid_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ return task_pid_nr(tsk);
```

```
+}
+
+static inline void set_task_vtgid(struct task_struct *tsk, pid_t nr)
+{
+}
+
+
+static inline pid_t task_pgrp_nr(struct task_struct *tsk)
+{
+ return tsk->signal->pgrp;
+}
+
+static inline pid_t task_pgrp_vnr(struct task_struct *tsk)
+{
+ return task_pgrp_nr(tsk);
+}
+
+static inline pid_t task_pgrp_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ return task_pgrp_nr(tsk);
+}
+
+static inline void set_task_vpgrp(struct task_struct *tsk, pid_t nr)
+{
+}
+
+
+static inline pid_t task_session_nr(struct task_struct *tsk)
+{
+ return tsk->signal->__session;
+}
+
+static inline pid_t task_session_vnr(struct task_struct *tsk)
+{
+ return task_session_nr(tsk);
+}
+
+static inline pid_t task_session_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ return task_session_nr(tsk);
+}
+
+static inline void set_task_vsession(struct task_struct *tsk, pid_t nr)
+{
+}
+
```

```
+
+static inline pid_t task_ppid_nr_ns(struct task_struct *tsk,
+  struct pid_namespace *ns)
+{
+ return rcu_dereference(tsk->real_parent)->tgid;
+}
+#else
+#endif
+
 /**
  * pid_alive - check that a task structure is not stale
  * @p: Task structure to be checked.
```

_____

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