
Subject: [PATCH 9/20] Helpers to obtain pid numbers
Posted by [Pavel Emelianov](#) on Tue, 07 Aug 2007 09:29:52 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 Emelyanov <xemul@openvz.org>

```
include/linux/pid.h | 24 ++++++++
include/linux/sched.h | 108 ++++++++++++++++++++++++++++++++++++++-----
kernel/pid.c | 13 ++++++
3 files changed, 134 insertions(+), 11 deletions(-)
```

```
--- ./include/linux/pid.h.ve9 2007-08-06 12:44:56.000000000 +0400
+++ ./include/linux/pid.h 2007-08-06 12:44:56.000000000 +0400
@@ -104,11 +104,33 @@ extern struct pid *find_ge_pid(int nr);
extern struct pid *alloc_pid(struct pid_namespace *ns);
extern void FASTCALL(free_pid(struct pid *pid));
```

```
+/*
+ * 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
+ */
+
+static inline pid_t pid_nr(struct pid *pid)
+{
+    pid_t nr = 0;
+    if (pid)
+    - nr = pid->nr;
+    + nr = pid->numbers[0].nr;
+    + return nr;
+}
+
+pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns);
```

```

+
+static inline pid_t pid_vnr(struct pid *pid)
+{
+ pid_t nr = 0;
+ if (pid)
+ nr = pid->numbers[pid->level].nr;
+ return nr;
+ }

--- ./include/linux/sched.h.ve9 2007-08-06 12:44:54.000000000 +0400
+++ ./include/linux/sched.h 2007-08-06 12:44:56.000000000 +0400
@@ -1221,16 +1221,6 @@ static inline int rt_task(struct task_st
+ return rt_prio(p->prio);
+ }

-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;
@@ -1256,6 +1246,104 @@ 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 sense 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
+ */
+
+static inline pid_t task_pid_nr(struct task_struct *tsk)
+{
+ return tsk->pid;
+}
+
+static inline pid_t task_pid_nr_ns(struct task_struct *tsk,
+ struct pid_namespace *ns)
+{
+ return pid_nr_ns(task_pid(tsk), ns);
+}
+
+static inline pid_t task_pid_vnr(struct task_struct *tsk)
+{
+ return pid_vnr(task_pid(tsk));
+}
+
+
+static inline pid_t task_tgid_nr(struct task_struct *tsk)
+{
+ return tsk->tgid;
+}
+
+static inline pid_t task_tgid_nr_ns(struct task_struct *tsk,
+ struct pid_namespace *ns)
+{
+ return pid_nr_ns(task_tgid(tsk), ns);
+}
+
+static inline pid_t task_tgid_vnr(struct task_struct *tsk)
+{
+ return pid_vnr(task_tgid(tsk));
+}
+
+
+static inline pid_t task_pgrp_nr(struct task_struct *tsk)
+{
+ return tsk->signal->pgrp;
+}
+
+static inline pid_t task_pgrp_nr_ns(struct task_struct *tsk,
```

```

+ struct pid_namespace *ns)
+{
+ return pid_nr_ns(task_pgrp(tsk), ns);
+}
+
+static inline pid_t task_pgrp_vnr(struct task_struct *tsk)
+{
+ return pid_vnr(task_pgrp(tsk));
+}
+
+
+static inline pid_t task_session_nr(struct task_struct *tsk)
+{
+ return tsk->signal->__session;
+}
+
+static inline pid_t task_session_nr_ns(struct task_struct *tsk,
+ struct pid_namespace *ns)
+{
+ return pid_nr_ns(task_session(tsk), ns);
+}
+
+static inline pid_t task_session_vnr(struct task_struct *tsk)
+{
+ return pid_vnr(task_session(tsk));
+}
+
+
+static inline pid_t task_ppid_nr_ns(struct task_struct *tsk,
+ struct pid_namespace *ns)
+{
+ return pid_nr_ns(task_pid(rcu_dereference(tsk->real_parent)), ns);
+}
+
+/**
+ * pid_alive - check that a task structure is not stale
+ * @p: Task structure to be checked.
+--- ./kernel/pid.c.ve9 2007-08-06 12:44:56.000000000 +0400
+++ ./kernel/pid.c 2007-08-06 12:44:56.000000000 +0400
@@ -380,6 +380,19 @@ struct pid *find_get_pid(pid_t nr)
    return pid;
}

+pid_t pid_nr_ns(struct pid *pid, struct pid_namespace *ns)
+{
+ struct upid *upid;
+ pid_t nr = 0;
+

```

```
+ if (pid && ns->level <= pid->level) {
+ upid = &pid->numbers[ns->level];
+ if (upid->ns == ns)
+ nr = upid->nr;
+ }
+ return nr;
+}
+
/*
 * Used by proc to find the first pid that is greater then or equal to nr.
 *
```
