Subject: [PATCH 1/4] Pidns: make full use of xxx_vnr() calls
Posted by Pavel Emelianov on Mon, 28 Jan 2008 15:05:37 GMT
View Forum Message <> Reply to Message

Some time ago the xxx_vnr() calls (e.g. pid_vnr or find_task_by_vpid)
were _all_ converted to operate on the current pid namespace. After
this each call like xxx_nr_ns(foo, current->nsproxy->pid_ns) is
nothing but a xxx_vnr(foo) one.

Switch all the xxx_nr_ns() callers to use the xxx_vnr() calls where
appropriate.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Reviewed-by: Oleg Nesterov <oleg@tv-sign.ru>

```
 fs/fcntl.c      |    2 +-
 fs/locks.c      |    5 ++---
 ipc/mqueue.c    |    3 +--
 kernel/exit.c   |    6 +++---
 kernel/fork.c   |    8 +-------
 kernel/sys.c    |    7 ++-----
 kernel/sysctl.c |    2 +-
 kernel/timer.c  |    2 +-
 8 files changed, 12 insertions(+), 23 deletions(-)

diff --git a/fs/fcntl.c b/fs/fcntl.c
index 603732a..e632da7 100644
--- a/fs/fcntl.c
+++ b/fs/fcntl.c
@@ -309,7 +309,7 @@ pid_t f_getown(struct file *filp)
 {
  pid_t pid;
  read_lock(&filp->f_owner.lock);
- pid = pid_nr_ns(filp->f_owner.pid, current->nsproxy->pid_ns);
+ pid = pid_vnr(filp->f_owner.pid);
  if (filp->f_owner.pid_type == PIDTYPE_PGID)
   pid = -pid;
  read_unlock(&filp->f_owner.lock);
diff --git a/fs/locks.c b/fs/locks.c
index bc3ceb3..5b8bfd8 100644
--- a/fs/locks.c
+++ b/fs/locks.c
@@ -685,8 +685,7 @@ posix_test_lock(struct file *filp, struct file_lock *fl)
  if (cfl) {
   __locks_copy_lock(fl, cfl);
   if (cfl->fl_nspid)
```

```
-   fl->fl_pid = pid_nr_ns(cfl->fl_nspid,
-      task_active_pid_ns(current));
+   fl->fl_pid = pid_vnr(cfl->fl_nspid);
  } else
   fl->fl_type = F_UNLCK;
  unlock_kernel();
@@ -2098,7 +2097,7 @@ static void lock_get_status(struct seq_file *f, struct file_lock *fl,
  unsigned int fl_pid;

  if (fl->fl_nspid)
-  fl_pid = pid_nr_ns(fl->fl_nspid, task_active_pid_ns(current));
+  fl_pid = pid_vnr(fl->fl_nspid);
  else
   fl_pid = fl->fl_pid;

diff --git a/ipc/mqueue.c b/ipc/mqueue.c
index 7d1b8aa..eb3f489 100644
--- a/ipc/mqueue.c
+++ b/ipc/mqueue.c
@@ -332,8 +332,7 @@ static ssize_t mqueue_read_file(struct file *filp, char __user *u_data,
    (info->notify_owner &&
     info->notify.sigev_notify == SIGEV_SIGNAL) ?
     info->notify.sigev_signo : 0,
-   pid_nr_ns(info->notify_owner,
-    current->nsproxy->pid_ns));
+   pid_vnr(info->notify_owner));
  spin_unlock(&info->lock);
  buffer[sizeof(buffer)-1] = '\0';
  slen = strlen(buffer)+1;
diff --git a/kernel/exit.c b/kernel/exit.c
index 56098de..44cc032 100644
--- a/kernel/exit.c
+++ b/kernel/exit.c
@@ -1170,7 +1170,7 @@ static int wait_task_zombie(struct task_struct *p, int noreap,
 {
  unsigned long state;
  int retval, status, traced;
- pid_t pid = task_pid_nr_ns(p, current->nsproxy->pid_ns);
+ pid_t pid = task_pid_vnr(p);

  if (unlikely(noreap)) {
   uid_t uid = p->uid;
@@ -1365,7 +1365,7 @@ unlock_sig:
   * possibly take page faults for user memory.
   */
  get_task_struct(p);
- pid = task_pid_nr_ns(p, current->nsproxy->pid_ns);
+ pid = task_pid_vnr(p);
```

```
  why = (p->ptrace & PT_PTRACED) ? CLD_TRAPPED : CLD_STOPPED;
  read_unlock(&tasklist_lock);

@@ -1424,7 +1424,7 @@ static int wait_task_continued(struct task_struct *p, int noreap,
   p->signal->flags &= ~SIGNAL_STOP_CONTINUED;
  spin_unlock_irq(&p->sighand->siglock);

- pid = task_pid_nr_ns(p, current->nsproxy->pid_ns);
+ pid = task_pid_vnr(p);
  uid = p->uid;
  get_task_struct(p);
  read_unlock(&tasklist_lock);
diff --git a/kernel/fork.c b/kernel/fork.c
index 16d98d4..40c87cc 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -1460,13 +1460,7 @@ long do_fork(unsigned long clone_flags,
  if (!IS_ERR(p)) {
   struct completion vfork;

-  /*
-   * this is enough to call pid_nr_ns here, but this if
-   * improves optimisation of regular fork()
-   */
-  nr = (clone_flags & CLONE_NEWPID) ?
-   task_pid_nr_ns(p, current->nsproxy->pid_ns) :
-   task_pid_vnr(p);
+  nr = task_pid_vnr(p);

   if (clone_flags & CLONE_PARENT_SETTID)
    put_user(nr, parent_tidptr);
diff --git a/kernel/sys.c b/kernel/sys.c
index 5a61f80..a626116 100644
--- a/kernel/sys.c
+++ b/kernel/sys.c
@@ -991,17 +991,14 @@ asmlinkage long sys_getpgid(pid_t pid)
  else {
   int retval;
   struct task_struct *p;
-  struct pid_namespace *ns;
-
-  ns = current->nsproxy->pid_ns;

   read_lock(&tasklist_lock);
-  p = find_task_by_pid_ns(pid, ns);
+  p = find_task_by_vpid(pid);
   retval = -ESRCH;
   if (p) {
```

```
    retval = security_task_getpgid(p);
    if (!retval)
-    retval = task_pgrp_nr_ns(p, ns);
+    retval = task_pgrp_vnr(p);
   }
   read_unlock(&tasklist_lock);
   return retval;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 62e971d..3b63246 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -2477,7 +2477,7 @@ static int proc_do_cad_pid(struct ctl_table *table, int write, struct file
*filp
  pid_t tmp;
  int r;

- tmp = pid_nr_ns(cad_pid, current->nsproxy->pid_ns);
+ tmp = pid_vnr(cad_pid);

  r = __do_proc_dointvec(&tmp, table, write, filp, buffer,
          lenp, ppos, NULL, NULL);
diff --git a/kernel/timer.c b/kernel/timer.c
index e8662eb..4f9827b 100644
--- a/kernel/timer.c
+++ b/kernel/timer.c
@@ -979,7 +979,7 @@ asmlinkage long sys_getppid(void)
  int pid;

  rcu_read_lock();
- pid = task_tgid_nr_ns(current->real_parent, current->nsproxy->pid_ns);
+ pid = task_tgid_vnr(current->real_parent);
  rcu_read_unlock();

  return pid;
--
1.5.3.4
```