Subject: [PATCH 12/12] Deprecate the find_task_by_pid
Posted by Pavel Emelianov on Tue, 29 Jan 2008 14:09:55 GMT
View Forum Message <> Reply to Message

There are some places that are known to operate on tasks'
global pids only:

* the rest_init() call (called on boot)
* the kgdb's getthread
* the create_kthread() (since the kthread is run in init ns)

So use the find_task_by_pid_ns(..., &init_pid_ns) there
and schedule the find_task_by_pid for removal.

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

```
---
 Documentation/feature-removal-schedule.txt |   19 +++++++++++++++++++
 include/linux/sched.h                      |    2 +-
 init/main.c                                |    2 +-
 kernel/kgdb.c                              |    2 +-
 kernel/kthread.c                           |    2 +-
 5 files changed, 23 insertions(+), 4 deletions(-)

diff --git a/Documentation/feature-removal-schedule.txt
b/Documentation/feature-removal-schedule.txt
index be6c2db..1e6960d 100644
--- a/Documentation/feature-removal-schedule.txt
+++ b/Documentation/feature-removal-schedule.txt
@@ -264,3 +264,22 @@ Why: These drivers are superseded by i810fb, intelfb and savagefb.
 Who: Jean Delvare <khali@linux-fr.org>

 ---------------------------
+
+What: find_task_by_pid
+When: 2.6.26
+Why: With pid namespaces, calling this funciton will return the
+ wrong task when called from inside a namespace.
+
+ The best way to save a task pid and find a task by this
+ pid later, is to find this task's struct pid pointer (or get
+ it directly from the task) and call pid_task() later.
+
+ If someone really needs to get a task by its pid_t, then
+ he most likely needs the find_task_by_vpid() to get the
+ task from the same namespace as the current task is in, but
+ this may be not so in general.
+
```

```
+Who: Pavel Emelyanov <xemul@openvz.org>
+
+---------------------------
+
diff --git a/include/linux/sched.h b/include/linux/sched.h
index 198659b..ccb9bba 100644
--- a/include/linux/sched.h
+++ b/include/linux/sched.h
@@ -1617,7 +1617,7 @@ 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);

-extern struct task_struct *find_task_by_pid(pid_t nr);
+extern struct task_struct *find_task_by_pid(pid_t nr) __deprecated;
 extern struct task_struct *find_task_by_vpid(pid_t nr);
 extern struct task_struct *find_task_by_pid_ns(pid_t nr,
   struct pid_namespace *ns);
diff --git a/init/main.c b/init/main.c
index e702632..38a16ba 100644
--- a/init/main.c
+++ b/init/main.c
@@ -437,7 +437,7 @@ static void noinline __init_refok rest_init(void)
  kernel_thread(kernel_init, NULL, CLONE_FS | CLONE_SIGHAND);
  numa_default_policy();
  pid = kernel_thread(kthreadd, NULL, CLONE_FS | CLONE_FILES);
- kthreadd_task = find_task_by_pid(pid);
+ kthreadd_task = find_task_by_pid_ns(pid, &init_pid_ns);
  unlock_kernel();

  /*
diff --git a/kernel/kgdb.c b/kernel/kgdb.c
index 87b0463..6de0fd0 100644
--- a/kernel/kgdb.c
+++ b/kernel/kgdb.c
@@ -621,7 +621,7 @@ static struct task_struct *getthread(struct pt_regs *regs, int tid)
  if (!tid)
   return NULL;

- return find_task_by_pid(tid);
+ return find_task_by_pid_ns(tid, &init_pid_ns);
 }

 #ifdef CONFIG_SMP
diff --git a/kernel/kthread.c b/kernel/kthread.c
index 0ac8878..b9caa46 100644
--- a/kernel/kthread.c
+++ b/kernel/kthread.c
@@ -99,7 +99,7 @@ static void create_kthread(struct kthread_create_info *create)
```

```
   struct sched_param param = { .sched_priority = 0 };
   wait_for_completion(&create->started);
   read_lock(&tasklist_lock);
-  create->result = find_task_by_pid(pid);
+  create->result = find_task_by_pid_ns(pid, &init_pid_ns);
   read_unlock(&tasklist_lock);
   /*
    * root may have changed our (kthreadd's) priority or CPU mask.
--
1.5.3.4
```