Subject: [PATCH 13/20] Miscelaneous preparations for pid namespaces
Posted by Pavel Emelianov on Tue, 07 Aug 2007 09:30:00 GMT
View Forum Message <> Reply to Message

* remove pid.h from pid_namespaces.h;
* rework is_(container|global)_init;
* optimize (get|put)_pid_ns for init_pid_ns;
* declare task_child_reaper to return actual reaper.

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

---

 include/linux/pid_namespace.h | 10 ++++++----
 include/linux/sched.h | 12 +++++-------
 kernel/pid.c | 22 ++++++++++++++++++++--------
 3 files changed, 25 insertions(+), 19 deletions(-)

--- ./include/linux/pid_namespace.h.ve17 2007-08-06 12:44:57.000000000 +0400
+++ ./include/linux/pid_namespace.h 2007-08-06 12:44:57.000000000 +0400
@@ -4,7 +4,6 @@
 #include <linux/sched.h>
 #include <linux/mm.h>
 #include <linux/threads.h>
-#include <linux/pid.h>
 #include <linux/nsproxy.h>
 #include <linux/kref.h>

@@ -30,7 +29,8 @@ extern struct pid_namespace init_pid_ns;

 static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
 {
- kref_get(&ns->kref);
+ if (ns != &init_pid_ns)
+ kref_get(&ns->kref);
  return ns;
 }

@@ -39,7 +39,8 @@ extern void free_pid_ns(struct kref *kre

 static inline void put_pid_ns(struct pid_namespace *ns)
 {
- kref_put(&ns->kref, free_pid_ns);
+ if (ns != &init_pid_ns)
+ kref_put(&ns->kref, free_pid_ns);
 }

 static inline struct pid_namespace *task_active_pid_ns(struct task_struct *tsk)

```
@@ -49,7 +50,8 @@ static inline struct pid_namespace *task

 static inline struct task_struct *task_child_reaper(struct task_struct *tsk)
 {
- return init_pid_ns.child_reaper;
+ BUG_ON(tsk != current);
+ return tsk->nsproxy->pid_ns->child_reaper;
 }

 #endif /* _LINUX_PID_NS_H */
--- ./include/linux/sched.h.ve17 2007-08-06 12:44:57.000000000 +0400
+++ ./include/linux/sched.h 2007-08-06 12:44:57.000000000 +0400
@@ -1362,19 +1362,17 @@ static inline int pid_alive(struct task_
  * @tsk: Task structure to be checked.
  *
  * Check if a task structure is the first user space task the kernel created.
- *
- * TODO: We should inline this function after some cleanups in pid_namespace.h
  */
-extern int is_global_init(struct task_struct *tsk);
+static inline int is_global_init(struct task_struct *tsk)
+{
+ return tsk->pid == 1;
+}

 /*
  * is_container_init:
  * check whether in the task is init in its own pid namespace.
  */
-static inline int is_container_init(struct task_struct *tsk)
-{
- return tsk->pid == 1;
-}
+extern int is_container_init(struct task_struct *tsk);

 extern struct pid *cad_pid;

--- ./kernel/pid.c.ve17 2007-08-06 12:44:57.000000000 +0400
+++ ./kernel/pid.c 2007-08-06 12:44:57.000000000 +0400
@@ -73,11 +73,20 @@ struct pid_namespace init_pid_ns = {
 };
 EXPORT_SYMBOL(init_pid_ns);

-int is_global_init(struct task_struct *tsk)
+int is_container_init(struct task_struct *tsk)
 {
- return tsk == init_pid_ns.child_reaper;
+ int ret = 0;
```

```
+ struct pid *pid;
+
+ rcu_read_lock();
+ pid = task_pid(tsk);
+ if (pid != NULL && pid->numbers[pid->level].nr == 1)
+  ret = 1;
+ rcu_read_unlock();
+
+ return ret;
 }
-EXPORT_SYMBOL(is_global_init);
+EXPORT_SYMBOL(is_container_init);

 /*
  * Note: disable interrupts while the pidmap_lock is held as an
@@ -193,8 +202,7 @@ fastcall void put_pid(struct pid *pid)
  if ((atomic_read(&pid->count) == 1) ||
       atomic_dec_and_test(&pid->count)) {
   kmem_cache_free(ns->pid_cachep, pid);
-  if (ns != &init_pid_ns)
-   put_pid_ns(ns);
+  put_pid_ns(ns);
  }
 }
 EXPORT_SYMBOL_GPL(put_pid);
@@ -244,9 +252,7 @@ struct pid *alloc_pid(struct pid_namespa
   tmp = tmp->parent;
  }

- if (ns != &init_pid_ns)
-  get_pid_ns(ns);
-
+ get_pid_ns(ns);
  pid->level = ns->level;
  pid->nr = pid->numbers[0].nr;
  atomic_set(&pid->count, 1);
```