Subject: [PATCH 4/4] pid: Move all of the pid_namespace logic into copy_pid_ns.
Posted by ebiederm on Wed, 12 Dec 2007 13:33:27 GMT
View Forum Message <> Reply to Message

Currently we have a bunch of pid namespace logic unnecessarily
scattered through fork, this patch moves it into copy_pid_ns
where it is easy to find and review.

To do this the prototype of copy_pid_ns is changed to take
a task parameter and the pid passed into copy_process
is passed in the task_pid field of that task.

After this cleanup things are actually clean enough we can contemplate
what an unshare of the pid namespace would mean.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 include/linux/pid_namespace.h |    5 +++--
 kernel/fork.c                 |   16 +++++++-----------
 kernel/nsproxy.c              |    2 +-
 kernel/pid_namespace.c        |   31 +++++++++++++++++++++++++++++++---
 4 files changed, 37 insertions(+), 17 deletions(-)

diff --git a/include/linux/pid_namespace.h b/include/linux/pid_namespace.h
index 2e67033..8e2e346 100644
--- a/include/linux/pid_namespace.h
+++ b/include/linux/pid_namespace.h
@@ -37,7 +37,7 @@ static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
  return ns;
 }

-extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *ns);
+extern struct pid_namespace *copy_pid_ns(unsigned long flags, struct task_struct *tsk);
 extern void free_pid_ns(struct kref *kref);
 extern void zap_pid_ns_processes(struct pid_namespace *pid_ns);

@@ -56,8 +56,9 @@ static inline struct pid_namespace *get_pid_ns(struct pid_namespace *ns)
 }

 static inline struct pid_namespace *
-copy_pid_ns(unsigned long flags, struct pid_namespace *ns)
+copy_pid_ns(unsigned long flags, struct task_struct *task)
 {
+ struct pid_namespace *ns = task->nsproxy->pid_ns;
  if (flags & CLONE_NEWPID)
   ns = ERR_PTR(-EINVAL);
  return ns;
diff --git a/kernel/fork.c b/kernel/fork.c

```
index 20d26a5..bbd8bcd 100644
--- a/kernel/fork.c
+++ b/kernel/fork.c
@@ -50,7 +50,6 @@
 #include <linux/taskstats_kern.h>
 #include <linux/random.h>
 #include <linux/tty.h>
-#include <linux/proc_fs.h>
 #include <linux/memcontrol.h>

 #include <asm/pgtable.h>
@@ -1013,6 +1012,9 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  if (!p)
   goto fork_out;

+ /* Remember the pid */
+ p->pids[PIDTYPE_PID].pid = pid;
+
  rt_mutex_init_task(p);

 #ifdef CONFIG_TRACE_IRQFLAGS
@@ -1162,17 +1164,12 @@ static struct task_struct *copy_process(unsigned long clone_flags,
  if (retval)
   goto bad_fork_cleanup_namespaces;

- if (pid != &init_struct_pid) {
+ pid = task_pid(p);
+ if (!pid) {
   retval = -ENOMEM;
   pid = alloc_pid(p->nsproxy->pid_ns);
   if (!pid)
    goto bad_fork_cleanup_namespaces;
-
-  if (clone_flags & CLONE_NEWPID) {
-   retval = pid_ns_prepare_proc(p->nsproxy->pid_ns);
-   if (retval < 0)
-    goto bad_fork_free_pid;
-  }
  }

  p->pid = pid_nr(pid);
@@ -1304,9 +1301,6 @@ static struct task_struct *copy_process(unsigned long clone_flags,
    __ptrace_link(p, current->parent);

   if (thread_group_leader(p)) {
-   if (clone_flags & CLONE_NEWPID)
-    p->nsproxy->pid_ns->child_reaper = pid;
-
```

```
    p->signal->leader_pid = pid;
    p->signal->tty = current->signal->tty;
    set_task_pgrp(p, task_pgrp_nr(current));
diff --git a/kernel/nsproxy.c b/kernel/nsproxy.c
index f5d332c..7b6d2dc 100644
--- a/kernel/nsproxy.c
+++ b/kernel/nsproxy.c
@@ -75,7 +75,7 @@ static struct nsproxy *create_new_namespaces(unsigned long flags,
   goto out_ipc;
  }

- new_nsp->pid_ns = copy_pid_ns(flags, task_active_pid_ns(tsk));
+ new_nsp->pid_ns = copy_pid_ns(flags, tsk);
  if (IS_ERR(new_nsp->pid_ns)) {
   err = PTR_ERR(new_nsp->pid_ns);
   goto out_pid;
diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c
index 6d792b6..739a72b 100644
--- a/kernel/pid_namespace.c
+++ b/kernel/pid_namespace.c
@@ -12,6 +12,7 @@
 #include <linux/pid_namespace.h>
 #include <linux/syscalls.h>
 #include <linux/err.h>
+#include <linux/proc_fs.h>

 #define BITS_PER_PAGE  (PAGE_SIZE*8)

@@ -115,9 +116,12 @@ static void destroy_pid_namespace(struct pid_namespace *ns)
  kmem_cache_free(pid_ns_cachep, ns);
 }

-struct pid_namespace *copy_pid_ns(unsigned long flags, struct pid_namespace *old_ns)
+struct pid_namespace *copy_pid_ns(unsigned long flags, struct task_struct *tsk)
 {
+ struct pid_namespace *old_ns = tsk->nsproxy->pid_ns;
  struct pid_namespace *new_ns;
+ struct pid *pid = NULL;
+ int err;

  BUG_ON(!old_ns);
  new_ns = get_pid_ns(old_ns);
@@ -129,13 +133,34 @@ struct pid_namespace *copy_pid_ns(unsigned long flags, struct
pid_namespace *old
   goto out_put;

  new_ns = create_pid_namespace(old_ns->level + 1);
- if (!IS_ERR(new_ns))
```

```
-  new_ns->parent = get_pid_ns(old_ns);
+ if (IS_ERR(new_ns))
+  goto out_put;
+
+ new_ns->parent = get_pid_ns(old_ns);
+ pid = task_pid(tsk);
+ if (!pid) {
+  err = -ENOMEM;
+  tsk->pids[PIDTYPE_PID].pid = pid = alloc_pid(new_ns);
+  if (!pid)
+   goto out_put_new;
+ }
+
+ new_ns->child_reaper = pid;
+ err = pid_ns_prepare_proc(new_ns);
+ if (err)
+  goto out_put_new;

 out_put:
  put_pid_ns(old_ns);
 out:
  return new_ns;
+
+out_put_new:
+ if (pid)
+  free_pid(pid);
+ put_pid_ns(new_ns);
+ new_ns = ERR_PTR(err);
+ goto out_put;
 }

 void free_pid_ns(struct kref *kref)
--
1.5.3.rc6.17.g1911
```