
Subject: [PATCH 2/3] pidns: Call pid_ns_prepare_proc from create_pid_namespace
Posted by [Daniel Lezcano](#) on Mon, 14 Feb 2011 19:16:02 GMT
[View Forum Message](#) <> [Reply to Message](#)

From: Eric W. Biederman <ebiederm@xmission.com>

Reorganize proc_get_sb so it can be called before the struct pid
of the first process is allocated.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

Signed-off-by: Daniel Lezcano <daniel.lezcano@free.fr>

```
fs/proc/root.c      | 25 ++++++-----  
kernel/fork.c        |  6 -----  
kernel/pid_namespace.c | 11 ++++++++--  
3 files changed, 16 insertions(+), 26 deletions(-)
```

diff --git a/fs/proc/root.c b/fs/proc/root.c

index ef9fa8e..e5e2bfa 100644

--- a/fs/proc/root.c

+++ b/fs/proc/root.c

```
@@ -43,17 +43,6 @@ static struct dentry *proc_mount(struct file_system_type *fs_type,  
    struct pid_namespace *ns;  
    struct proc_inode *ei;
```

```
- if (proc_mnt) {  
- /* Seed the root directory with a pid so it doesn't need  
- * to be special in base.c. I would do this earlier but  
- * the only task alive when /proc is mounted the first time  
- * is the init_task and it doesn't have any pids.  
- */  
- ei = PROC_I(proc_mnt->mnt_sb->s_root->d_inode);  
- if (!ei->pid)  
- ei->pid = find_get_pid(1);  
- }
```

```
-  
if (flags & MS_KERNMOUNT)  
    ns = (struct pid_namespace *)data;  
else
```

```
@@ -71,16 +60,16 @@ static struct dentry *proc_mount(struct file_system_type *fs_type,  
    return ERR_PTR(err);  
}
```

```
- ei = PROC_I(sb->s_root->d_inode);  
- if (!ei->pid) {  
- rcu_read_lock();  
- ei->pid = get_pid(find_pid_ns(1, ns));  
- rcu_read_unlock();
```

```

- }
-
    sb->s_flags |= MS_ACTIVE;
}

+ ei = PROC_I(sb->s_root->d_inode);
+ if (!ei->pid) {
+   rcu_read_lock();
+   ei->pid = get_pid(find_pid_ns(1, ns));
+   rcu_read_unlock();
+ }
+
    return dget(sb->s_root);
}

```

diff --git a/kernel/fork.c b/kernel/fork.c

index c9f0784..e7a5907 100644

--- a/kernel/fork.c

+++ b/kernel/fork.c

```

@@ -1180,12 +1180,6 @@ static struct task_struct *copy_process(unsigned long clone_flags,
     pid = alloc_pid(p->nsproxy->pid_ns);
     if (!pid)
         goto bad_fork_cleanup_io;
-
- 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);

```

diff --git a/kernel/pid_namespace.c b/kernel/pid_namespace.c

index a5aff94..e9c9adc 100644

--- a/kernel/pid_namespace.c

+++ b/kernel/pid_namespace.c

@@ -14,6 +14,7 @@

#include <linux/err.h>

#include <linux/acct.h>

#include <linux/slab.h>

+#include <linux/proc_fs.h>

#define BITS_PER_PAGE (PAGE_SIZE*8)

```

@@ -72,7 +73,7 @@ static struct pid_namespace *create_pid_namespace(struct
pid_namespace *parent_p
{
    struct pid_namespace *ns;

```

```

    unsigned int level = parent_pid_ns->level + 1;
- int i;
+ int i, err = -ENOMEM;

    ns = kmem_cache_zalloc(pid_ns_cachep, GFP_KERNEL);
    if (ns == NULL)
@@ -96,14 +97,20 @@ static struct pid_namespace *create_pid_namespace(struct
pid_namespace *parent_p
    for (i = 1; i < PIDMAP_ENTRIES; i++)
        atomic_set(&ns->pidmap[i].nr_free, BITS_PER_PAGE);

+ err = pid_ns_prepare_proc(ns);
+ if (err)
+ goto out_put_parent_pid_ns;
+
    return ns;

+out_put_parent_pid_ns:
+ put_pid_ns(parent_pid_ns);
out_free_map:
    kfree(ns->pidmap[0].page);
out_free:
    kmem_cache_free(pid_ns_cachep, ns);
out:
- return ERR_PTR(-ENOMEM);
+ return ERR_PTR(err);
}

```

```
static void destroy_pid_namespace(struct pid_namespace *ns)
```

```
--
```

1.7.1

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
