

---

Subject: Re: [RFC, PATCH] handle the multi-threaded init's exit() properly  
Posted by [Oleg Nesterov](#) on Fri, 03 Aug 2007 18:26:31 GMT  
[View Forum Message](#) <> [Reply to Message](#)

---

On 08/02, Roland McGrath wrote:

>  
> This looks fine to me, though I don't know anything about the nsproxy bit.  
> Now that choose\_new\_parent is one trivial line, you might go on to get rid  
> of it and roll its one line into reparent\_thread.

OK, thanks. Please find the same patch + "kill one-liner reparent\_thread()" below.

[PATCH] handle the multi-threaded init's exit() properly

With or without this patch, multi-threaded init's are not fully supported, but do\_exit() is completely wrong. This becomes a real problem when we support pid namespaces.

1. do\_exit() panics when the main thread of /sbin/init exits. It should not until the whole thread group exits. Move the code below, under the "if (group\_dead)" check.

Note: this means that forget\_original\_parent() can use an already dead child\_reaper()'s task\_struct. This is OK for /sbin/init because

- do\_wait() from alive sub-thread still can reap a zombie, we iterate over all sub-thread's ->children lists
- do\_notify\_parent() will wakeup some alive sub-thread because it sends the group-wide signal

However, we should remove choose\_new\_parent()->BUG\_ON(reaper->exit\_state) for this.

2. We are playing games with ->nsproxy->pid\_ns. This code is bogus today, and it has to be changed anyway when we really support pid namespaces, just remove it.

Signed-off-by: Oleg Nesterov <[oleg@tv-sign.ru](mailto:oleg@tv-sign.ru)>

```
--- t/kernel/exit.c~MTINIT 2007-07-28 16:58:17.000000000 +0400
+++ t/kernel/exit.c 2007-08-03 18:05:52.000000000 +0400
@@ -601,17 +601,6 @@ static void exit_mm(struct task_struct *
     mmput(mm);
 }
```

```

-static inline void
-choose_new_parent(struct task_struct *p, struct task_struct *reaper)
-{
- /*
-  * Make sure we're not reparenting to ourselves and that
-  * the parent is not a zombie.
-  */
- BUG_ON(p == reaper || reaper->exit_state);
- p->real_parent = reaper;
-}
-
static void
reparent_thread(struct task_struct *p, struct task_struct *father, int traced)
{
@@ -719,7 +708,7 @@ forget_original_parent(struct task_struct *p, struct task_struct *father)
{
    if (father == p->real_parent) {
        /* reparent with a reaper, real father it's us */
-    choose_new_parent(p, reaper);
+    p->real_parent = reaper;
        reparent_thread(p, father, 0);
    } else {
        /* reparent ptraced task to its real parent */
@@ -740,7 +729,7 @@ forget_original_parent(struct task_struct *p, struct task_struct *father)
    }
    list_for_each_safe(_p, _n, &father->ptrace_children) {
        p = list_entry(_p, struct task_struct, ptrace_list);
-    choose_new_parent(p, reaper);
+    p->real_parent = reaper;
        reparent_thread(p, father, 1);
    }
}
@@ -895,6 +884,14 @@ static void check_stack_usage(void)
static inline void check_stack_usage(void) {}
#endif

+static inline void exit_child_reaper(struct task_struct *tsk)
+{
+ if (likely(tsk->group_leader != child_reaper(tsk)))
+ return;
+
+ panic("Attempted to kill init!");
+}
+
fastcall NORET_TYPE void do_exit(long code)
{
    struct task_struct *tsk = current;
@@ -908,13 +905,6 @@ fastcall NORET_TYPE void do_exit(long code)
}

```

```

    panic("Aiee, killing interrupt handler!");
    if (unlikely(!tsk->pid))
        panic("Attempted to kill the idle task!");
-   if (unlikely(tsk == child_reaper(tsk))) {
-   if (tsk->nsproxy->pid_ns != &init_pid_ns)
-   tsk->nsproxy->pid_ns->child_reaper = init_pid_ns.child_reaper;
-   else
-   panic("Attempted to kill init!");
- }
-

    if (unlikely(current->ptrace & PT_TRACE_EXIT)) {
        current->ptrace_message = code;
@@ -964,6 +954,7 @@ fastcall NORET_TYPE void do_exit(long co
    }
    group_dead = atomic_dec_and_test(&tsk->signal->live);
    if (group_dead) {
+   exit_child_reaper(tsk);
        hrtimer_cancel(&tsk->signal->real_timer);
        exit_itimers(tsk->signal);
    }

```

---

Containers mailing list  
Containers@lists.linux-foundation.org  
<https://lists.linux-foundation.org/mailman/listinfo/containers>

---