Subject: Re: [RFC][PATCH] Add child reaper to struct pspace
Posted by serue on Thu, 07 Sep 2006 01:39:54 GMT
View Forum Message <> Reply to Message

Quoting Sukadev Bhattiprolu (sukadev@us.ibm.com):
...
> Index: lx26-18-rc5/kernel/exit.c
> ===
> --- lx26-18-rc5.orig/kernel/exit.c 2006-09-06 17:04:03.000000000 -0700
> +++ lx26-18-rc5/kernel/exit.c 2006-09-06 17:18:47.000000000 -0700
> @@ -45,7 +45,6 @@
> #include <asm/mmu_context.h>
>
> extern void sem_exit (void);
> -extern struct task_struct *child_reaper;
>
> static void exit_mm(struct task_struct * tsk);
>
> @@ -267,7 +266,8 @@ static int has_stopped_jobs(int pgrp)
> }
>
> /**
> - * reparent_to_init - Reparent the calling kernel thread to the init task.
> + * reparent_to_init - Reparent the calling kernel thread to the init task
> + * of the pid space that the thread belongs to.
>   *
>   * If a kernel thread is launched as a result of a system call, or if
>   * it ever exits, it should generally reparent itself to init so that
> @@ -285,8 +285,8 @@ static void reparent_to_init(void)
>   ptrace_unlink(current);
>   /* Reparent to init */
>   remove_parent(current);
> - current->parent = child_reaper;
> - current->real_parent = child_reaper;
> + current->parent = current->pspace->child_reaper;
> + current->real_parent = current->pspace->child_reaper;
>   add_parent(current);
>
>   /* Set the exit signal to SIGCHLD so we signal init on exit */
> @@ -658,7 +658,8 @@ reparent_thread(struct task_struct *p, s
>   * When we die, we re-parent all our children.
>   * Try to give them to another thread in our thread
>   * group, and if no such member exists, give it to
> - * the global child reaper process (ie "init")
> + * the child reaper process (ie "init") in our pid
> + * space.
>   */
> static void

> forget_original_parent(struct task_struct *father, struct list_head *to_release)
> @@ -669,7 +670,7 @@ forget_original_parent(struct task_struc
>   do {
>   reaper = next_thread(reaper);
>   if (reaper == father) {
> -   reaper = child_reaper;
> +   reaper = father->pspace->child_reaper;
>   break;
>   }
>   } while (reaper->exit_state);
> @@ -857,7 +858,7 @@ fastcall NORET_TYPE void do_exit(long co
>   panic("Aiee, killing interrupt handler!");
>   if (unlikely(!tsk->pid))
>   panic("Attempted to kill the idle task!");
> - if (unlikely(tsk == child_reaper))
> + if (unlikely(tsk == tsk->pspace->child_reaper))
>   panic("Attempted to kill init!");

That becomes a little uncalled for now :)

Perhaps something more like

```
 if (unlikely(tsk == tsk->pspace->child_reaper)) {
  if (tsk->pspace != &init_pspace)
   tsk->pspace->child_reaper = init_pspace.child_reaper;
  else
   panic("Attempted to kill init!");
 }
```

It might be better yet to walk up the ancestor pspace tree, but I'm not
sure it's worth it at that point. There's no more hope of userspace
wanting to be informed, so we really just want task cleanup by the
kernel, so really any init task will do.

-serge

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers