Subject: Re: task->tgid conversion in fs/locks.c
Posted by Pavel Emelianov on Mon, 10 Sep 2007 07:24:36 GMT
View Forum Message <> Reply to Message

sukadev@us.ibm.com wrote:
> Pavel,
>
> I noticed that fcntl tests in LTP fail when LTP is run a child pid
> namespace. I just hacked up this quick patch and seems to fix the
> failures.
>
> Are you already working on this or do you want me to test and send
> this out for review.
>
> I have one concern that I could use some review/confirmation :-)
>
> Even if the main thread of a process exits before the child threads,
> the main thread will not be reaped until all threads exit. So, the
> 'task->group_leader' remains valid for the child threads until the
> last non-leader thread exits.
>
> IOW the call to task_tgid_vnr() is safe in locks_remove_posix() and
> locks_remove_flock().

Well, I missed this place deliberately. This should be converted into
struct pid * usage. I will do the appropriate patch soon.

> ---
> fs/locks.c | 14 +++++++-------
> 1 file changed, 7 insertions(+), 7 deletions(-)
>
> Index: 2.6.23-rc4-mm1/fs/locks.c
> ==========
> --- 2.6.23-rc4-mm1.orig/fs/locks.c 2007-09-05 12:24:30.000000000 -0700
> +++ 2.6.23-rc4-mm1/fs/locks.c 2007-09-07 09:14:47.000000000 -0700
> @@ -278,7 +278,7 @@ static int flock_make_lock(struct file *
> return -ENOMEM;
>
> fl->fl_file = filp;
> - fl->fl_pid = current->tgid;
> + fl->fl_pid = task_tgid_vnr(current);
> fl->fl_flags = FL_FLOCK;
> fl->fl_type = type;
> fl->fl_end = OFFSET_MAX;
> @@ -344,7 +344,7 @@ static int flock_to_posix_lock(struct fi
> return -EOVERFLOW;
>
> fl->fl_owner = current->files;

```
> - fl->fl_pid = current->tgid;
> + fl->fl_pid = task_tgid_vnr(current);
>   fl->fl_file = filp;
>   fl->fl_flags = FL_POSIX;
>   fl->fl_ops = NULL;
> @@ -390,7 +390,7 @@ static int flock64_to_posix_lock(struct
>    return -EOVERFLOW;
>
>   fl->fl_owner = current->files;
> - fl->fl_pid = current->tgid;
> + fl->fl_pid = task_tgid_vnr(current);
>   fl->fl_file = filp;
>   fl->fl_flags = FL_POSIX;
>   fl->fl_ops = NULL;
> @@ -446,7 +446,7 @@ static int lease_init(struct file *filp,
>    return -EINVAL;
>
>   fl->fl_owner = current->files;
> - fl->fl_pid = current->tgid;
> + fl->fl_pid = task_tgid_vnr(current);
>
>   fl->fl_file = filp;
>   fl->fl_flags = FL_LEASE;
> @@ -1091,7 +1091,7 @@ int locks_mandatory_area(int read_write,
>
>   locks_init_lock(&fl);
>   fl.fl_owner = current->files;
> - fl.fl_pid = current->tgid;
> + fl.fl_pid = task_tgid_vnr(current);
>   fl.fl_file = filp;
>   fl.fl_flags = FL_POSIX | FL_ACCESS;
>   if (filp && !(filp->f_flags & O_NONBLOCK))
> @@ -1963,7 +1963,7 @@ void locks_remove_posix(struct file *fil
>   lock.fl_start = 0;
>   lock.fl_end = OFFSET_MAX;
>   lock.fl_owner = owner;
> - lock.fl_pid = current->tgid;
> + lock.fl_pid = task_tgid_vnr(current);
>   lock.fl_file = filp;
>   lock.fl_ops = NULL;
>   lock.fl_lmops = NULL;
> @@ -1990,7 +1990,7 @@ void locks_remove_flock(struct file *fil
>
>   if (filp->f_op && filp->f_op->flock) {
>    struct file_lock fl = {
> -   .fl_pid = current->tgid,
> +   .fl_pid = task_tgid_vnr(current),
>     .fl_file = filp,
```

> .fl_flags = FL_FLOCK,
> .fl_type = F_UNLCK,
>

_____

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