Subject: Re: [PATCH] cgroup: fix a race condition in manipulating tsk->cg_list
Posted by Li Zefan on Thu, 17 Apr 2008 05:10:33 GMT
View Forum Message <> Reply to Message

Andrew Morton wrote:
> On Wed, 16 Apr 2008 21:17:34 -0700 "Paul Menage" <menage@google.com> wrote:
>
>> On Wed, Apr 16, 2008 at 9:11 PM, Andrew Morton
>> <akpm@linux-foundation.org> wrote:
>>> I don't fully understand the race. Both paths hold css_set_lock.
>>>
>>> Can you describe it in more detail please?
>> Task A starts exiting, passes the check for unlinking current->cg_list.
>
> So cgroup_exit() sees !list_empty(tsk->cg_list)
>

cgroup_exit() sees list_empty(tsk->cg_list), then cgroup_enable_task_cg_list()
links the task to the list, and then the task exited, so the list entry won't
get deleted.

> And the list_del() sets tsk->cg_list to LIST_POISON[12], which still means
> !list_empty(). Or we remove that debugging code and avoid writing to
> tsk->cg_list, and it _still_ is !list_empty().
>
>> Before it completely exits task B does the very first
>> cgroup_iter_begin() call (via reading a cgroups tasks file) which
>> links all tasks in to their css_set objects via tsk->cg_list.
>
> But it won't link this task, because it's !list_empty().
>
>> Then task A finishes exiting and is freed, but doesn't unlink from the cg_list.
>>
>>> afacit the task at *p could set PF_EXITING immediately after this code has
>>> tested PF_EXITING and then the task at *p could proceed until we hit the
>>> same race (whatever that is).
>> The important fact there is that the task sets PF_EXITING *before* it
>> checks whether it needs to unlink from current->cg_list.
>>
>> Paul
>
>