Subject: Re: [PATCH 1/15] Move exit_task_namespaces()
Posted by Pavel Emelianov on Mon, 06 Aug 2007 11:29:18 GMT
View Forum Message <> Reply to Message

Oleg Nesterov wrote:
> On 08/06, Pavel Emelyanov wrote:
>> Oleg Nesterov wrote:
>>> On 08/06, Pavel Emelyanov wrote:
>>>> Oleg Nesterov wrote:
>>>>> On 07/26, Pavel Emelyanov wrote:
>>>>>> The reason to release namespaces after reparenting is that when task
>>>>>> exits it may send a signal to its parent (SIGCHLD), but if the parent
>>>>>> has already exited its namespaces there will be no way to decide what
>>>>>> pid to dever to him - parent can be from different namespace.
>>>>> I almost forgot about this one...
>>>>>
>>>>> After reading the whole series, I can't understand the above explanation
>>>>> any longer. The parent can't be from different namespace, either we have
>>>>> another sub-thread, or we reparent the child to /sbin/init which should
>>>>> be from the same namespace.
>>>> If the child that is a new namespace's init is exiting its parent is from
>>>> the
>>>> different namespace.
>>> In that case it doesn't have childs. The were SIGKILL'ed before
>>> exit_notify().
>> It does not, but it's parent - does :)
>
> Yes. But in that case forget_original_parent() is no-op! This means that
> it is not needed to move exit_task_namepsace() after.
>
>>>> Moreover, we will probably want to implement "entering"
>>>> the pid namespace, so having tasks with parents from another namespace
>>>> will
>>>> be OK.
>>> Well. I saw this word "entering", but I don't know the meaning. Just
>>> curious,
>>> could you explain?
>> "Entering" means "moving task to arbitrary namespace"
>
> Heh. Very much nontrivial, good luck :) At least we need to grow ->numbers[],
> and if its pid was pinned...
>
>>> And, if an exiting task has a child which is already from another
>>> namespace,
>>> why can't we release our namespace before re-parenting? I guess I need to
>>> know what "entering" means to understand this...
>> One of the desired actions was the following:
>> 1. task X clones the new namespace with the child Y as this namespace's

>> init;
>> 2. task X waits for SIGCHILD to come informing that the namespace is dead.
>> In this scenario we need to set the Y's pid as it is seen from X's
>> namespace in siginfo.
>
> Yes sure. But again, how this is connected to "we should do exit_task_namespace()
> after forget_original_parent()" ?
>
> I guess I missed something stupid and simple...

In other words. Let task X live in init_pid_ns, task Y is his child and lives
int another namespace. task X and task Y both die. This will happen:

```
1. Task X call exit_task_namespaces()
   and sets its nsproxy to NULL
                                        1. Task Y is going to notify the
                                           parent (X) and dereferences its
                                           nsproxy -> OOPS
2. Task X reparents all its children
```

If we move the exit_task_namespace this will happen:

```
1. Task X reparents all its children
2. Task X call exit_task_namespaces()
   and sets its nsproxy to NULL
```

In such case is tasy Y will dereference the parent's nsproxy it will not
OOPS because either its parent will be not X already, or X's nsproxy is
not yet released.

> Oleg.
>
>