Subject: Re: pspace child_reaper
Posted by ebiederm on Tue, 29 Aug 2006 15:24:57 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

> Hello All,
>
> Eric, in your initial proof of concept on the pid namespace, you were
> defining a child_reaper per pid namespace.
>
> IMO, we can't use init_task as a child_reaper in a pid namespace because we
> will have pid collision which might result in a breakage of the init_task.

The kernel doesn't use init_task (The idle thread) once it starts
init. Reaping children is the job of pid == 1.

> Here are some questions on the model you intended to follow :
>
> Do you think we should have a child_reaper task per container ?
We have an init per container so yes.

> Could we use a kthread to do the job ?
Definitely not.

> Could that kthread be global to all pid namespace ?
Makes no sense.

> Any completely different idea on the topic ?
Init reaps the children, and I believe there are parts of user space
that depend on this. We shouldn't change that semantic.

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

Subject: Re: pspace child_reaper
Posted by Cedric Le Goater on Tue, 29 Aug 2006 15:40:59 GMT
View Forum Message <> Reply to Message

Eric W. Biederman wrote:
> Cedric Le Goater <clg@fr.ibm.com> writes:
>
>> Hello All,
>>

>> Eric, in your initial proof of concept on the pid namespace, you were
>> defining a child_reaper per pid namespace.
>>
>> IMO, we can't use init_task as a child_reaper in a pid namespace because we
>> will have pid collision which might result in a breakage of the init_task.
>
> The kernel doesn't use init_task (The idle thread) once it starts
> init. Reaping children is the job of pid == 1.

agree.

>> Here are some questions on the model you intended to follow :
>>
>> Do you think we should have a child_reaper task per container ?
> We have an init per container so yes.

hmm, have we always ? what if i don't start an /sbin/init process in my
newly created pid namespace or container. where do I collect all the SIGCHLD ?

>> Could we use a kthread to do the job ?
> Definitely not.

why ?

>> Could that kthread be global to all pid namespace ?
>
> Makes no sense.

if you don't have an init per container, we need to find someone for the job.

>> Any completely different idea on the topic ?
> Init reaps the children, and I believe there are parts of user space
> that depend on this. We shouldn't change that semantic.

IMHO, the only semantic i see is in the kernel, which needs someone to take
care of sigchld. /sbin/init is a very good candidate bc it collects sigchld
anyway and discards the ones it doesn't know about.

C.

____

Subject: Re: Re: pspace child_reaper

Posted by Cedric Le Goater on Wed, 30 Aug 2006 13:01:46 GMT
View Forum Message <> Reply to Message

Hello,

Roman Kagan wrote:

[...]

>> As for the per-container init process, the alternative to always
>> enforcing a separate init process for every container is to allow an
>> option of making the process which did the pidspace unshare (or is it
>> the parent of that process) masquerade as (pidspace=new_container, pid=1).
>
> There's no point enforcing a separate 'init' process in every container.
> The root of the process tree in a namespace has to be the child reaper
> for that namespace meaning that
>
> - it is immune to signals, ptracing, etc. from within the pidspace
> - every process in the pidspace is reparented to it once that process'
>   parent dies
> - when it dies the whole pidspace is termiated

That's how i feel also.

The key point here is that the process becoming the init of that pidspace
is immune to sigchlg : ignores them or garbage collects them or handles EINTR.

If we feel confortable with the above, let's bring back this question to a
user space issue : the process doing an unshare of this pidspace must
handle the sigchld one way or the other.

> These are the standard properties of pid == 1 in UNIX. If it happens to
> be (or execs) /sbin/init then indeed it'll sit in the background
> spawning the usual user processes when necessary, but it doesn't have to
> be. E.g. I've just run an FC5 machine with init=/usr/bin/python which
> is how your application container would probably look like (the result
> of 'import os; os.system("ps axf")' in python prompt):
>
```
>   PID TTY      STAT   TIME COMMAND
>     1 ?        S      0:00 /usr/bin/python
>     2 ?        SN     0:00 [ksoftirqd/0]
>     3 ?        S      0:00 [watchdog/0]
>     4 ?        S<     0:00 [events/0]
>     5 ?        S<     0:00 [khelper]
>     6 ?        S<     0:00 [kthread]
>     8 ?        S<     0:00  \_ [kblockd/0]
>     9 ?        S<     0:00  \_ [kacpid]
>    67 ?        S<     0:00  \_ [khubd]
>   122 ?       S      0:00  \_ [pdflush]
>   123 ?       S      0:00  \_ [pdflush]
>   125 ?       S<     0:00  \_ [aio/0]
>   212 ?       S<     0:00  \_ [kseriod]
>   282 ?       S<     0:00  \_ [kpsmoused]
>   303 ?       S<     0:00  \_ [scsi_eh_0]
>   124 ?       S      0:00 [kswapd0]
>   290 ?       Ss     0:00 /bin/nash /init
>   317 ?       S      0:00 [kjournald]
>   329 ?       R      0:00 sh -c ps axf
>   330 ?       R      0:00  \_ ps axf
```

yes

> so there's no fundamental difference between "system containers" and
> "application containers".

your example uses python which has a wait() loop sitting somewhere because
it needs to know how to handle processes, like any shell command
interpreter. but yes, it's something like this, with a process 1 knowing
how to handle sigchld.

thanks,

C.

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

Subject: Re: Re: pspace child_reaper
Posted by ebiederm on Wed, 30 Aug 2006 13:42:43 GMT
View Forum Message <> Reply to Message

Cedric Le Goater <clg@fr.ibm.com> writes:

> Hello,
>
> Roman Kagan wrote:
>
> [...]
>
>>> As for the per-container init process, the alternative to always
>>> enforcing a separate init process for every container is to allow an
>>> option of making the process which did the pidspace unshare (or is it
>>> the parent of that process) masquerade as (pidspace=new_container, pid=1).

>>
>> There's no point enforcing a separate 'init' process in every container.
>> The root of the process tree in a namespace has to be the child reaper
>> for that namespace meaning that
>>
>> - it is immune to signals, ptracing, etc. from within the pidspace
>> - every process in the pidspace is reparented to it once that process'
>>   parent dies
>> - when it dies the whole pidspace is termiated
>
> That's how i feel also.

Those sound like the correct semantics. Although terminating all of
it's children in a given pid namespace is an interesting semantic to
implement. But it seems to be the only sane one.   At least it
is better then the current version where the kernel exits if pid == 1
is terminated.

> The key point here is that the process becoming the init of that pidspace
> is immune to sigchlg : ignores them or garbage collects them or handles EINTR.
>
> If we feel confortable with the above, let's bring back this question to a
> user space issue : the process doing an unshare of this pidspace must
> handle the sigchld one way or the other.

Sounds good.

I'm not convinced an unshare of a pid namespace is a well defined
operation. But creating a new pid namespace at clone time certainly
is, and that is what replicates the python example.

Eric