Subject: Processes with multiple pid_t values
Posted by Sukadev Bhattiprolu on Fri, 08 Dec 2006 21:23:47 GMT
View Forum Message <> Reply to Message

A process that unshares its namespace gets a new pid_t in the child
namespace. Simlarly its process group and session leaders get new pid_ts
in the child namespace right ?

i.e do the following pid_ts look reasonable when process 1234 unshares
its pid namespace ?

PID PPID PGID SID

init pid ns 1234 1233 1230 1220

child pid ns 3 2 1 0

Assuming they are :-), we should expect find_pid() call with nr == 0, 1,
or 2 in the child pid namespace to also work right ?

But processes 1220, 1230, 1233 are entered into the hash table based on
their init pid ns values. And so the above find_pid() calls would not
find the process we want.

i.e some processes have two pid_ts and we want to find them using either
of the two values. The pid_hash table can obviously hash on one value.

We would need some serious changes to the pid_hash table to do this ???
(can we change the hash algorithm to generate a key based on all pid_ts
a process has ????)

Or should we just keep track of these special processes (4 per namespace
including the child reaper) in the namespace object - just like we treat
the child_reaper special ?

Then find_pid() would have to check the namespace object in addition to
hash table.

Appreciate any ideas/comments.

Suka

_____

Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:

> A process that unshares its namespace gets a new pid_t in the child
> namespace. Simlarly its process group and session leaders get new pid_ts
> in the child namespace right ?

So far the only place where switching to a new pid namespace has an
obvious answer is with sys clone. At that point the answer is:

PID == 1 PPID = 0 PGID = 1 SID = 1.

Just like /sbin/init gets.

>
> i.e do the following pid_ts look reasonable when process 1234 unshares
> its pid namespace ?
>
>
>   PID PPID PGID SID
>
> init pid ns 1234 1233 1230 1220
>
> child pid ns 3 2 1 0
>
> Assuming they are :-), we should expect find_pid() call with nr == 0, 1,
> or 2 in the child pid namespace to also work right ?

find_pid with nr == 0 will always fail.

> But processes 1220, 1230, 1233 are entered into the hash table based on
> their init pid ns values. And so the above find_pid() calls would not
> find the process we want.
>
> i.e some processes have two pid_ts and we want to find them using either
> of the two values. The pid_hash table can obviously hash on one value.
>
> We would need some serious changes to the pid_hash table to do this ???
> (can we change the hash algorithm to generate a key based on all pid_ts
> a process has ????)

You just enter a struct pid into the hash table based upon each pid_t value
assigned to it. You make the key comparison (namespace, pid_t) == (namespace, pid_t)
when doing the hash lookup.

> Or should we just keep track of these special processes (4 per namespace

> including the child reaper) in the namespace object - just like we treat
> the child_reaper special ?
>
> Then find_pid() would have to check the namespace object in addition to
> hash table.

Assume that we will have pid namespaces nested in pid namespaces 2 or 3 levels
deep.

Understand what a session id, a process group id, and a pid are and how
they are normally assigned please.

Don't even consider unshare of the pid namespace until you can see how to handle
the simpler clone case.

Eric

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

Subject: Re: Processes with multiple pid_t values
Posted by ebiederm on Tue, 12 Dec 2006 04:25:23 GMT
View Forum Message <> Reply to Message

Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:

> A process that unshares its namespace gets a new pid_t in the child
> namespace. Simlarly its process group and session leaders get new pid_ts
> in the child namespace right ?
>
> i.e do the following pid_ts look reasonable when process 1234 unshares
> its pid namespace ?
>
>
>   PID PPID PGID SID
>
> init pid ns 1234 1233 1230 1220
>
> child pid ns 3 2 1 0

A slightly more complete answer.

A pid that cannot be represented in the current pid namespace should be
0.

pid 1 is very special and in the case of a clone should definitely

be the first pid in the namespace.

In the case of an unshare pid == 1 is probably the process that does
the unshare, and it's children all show up in the child namespace.

Any other pids should simply be allocated with the pid allocator if they
do not fall into the pid namespace.

> Assuming they are :-), we should expect find_pid() call with nr == 0, 1,
> or 2 in the child pid namespace to also work right ?
>
> But processes 1220, 1230, 1233 are entered into the hash table based on
> their init pid ns values. And so the above find_pid() calls would not
> find the process we want.

My expectation is that we will have a link entry that points to the
real struct pid. Or possibly a different data structure at that point.
The pid hash table does not scale well to very large numbers of pids.

> i.e some processes have two pid_ts and we want to find them using either
> of the two values. The pid_hash table can obviously hash on one value.
>
> We would need some serious changes to the pid_hash table to do this ???
> (can we change the hash algorithm to generate a key based on all pid_ts
> a process has ????)

The key would need to be pid_namespace, pid_t. We just need a few tweaks
to the lookup algorithm. It's 5-10 line patch most likely.

> Or should we just keep track of these special processes (4 per namespace
> including the child reaper) in the namespace object - just like we treat
> the child_reaper special ?

It doesn't look hard to allow pids to appear in several namespaces,
we need at least one pid to appear in multiple pid namespaces, so
it makes sense to handle that case. With struct pid and the helper
functions currently defined it is not hard to allow for all pids to
be in several namespaces.

So it is my intention that all processes will have a pid in every parent
pid namespace as well as a pid in their current pid namespace.

Eric

Subject: Re: Processes with multiple pid_t values
Posted by serue on Tue, 12 Dec 2006 16:04:55 GMT
View Forum Message <> Reply to Message

Quoting Eric W. Biederman (ebiederm@xmission.com):
> Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:
>
> > A process that unshares its namespace gets a new pid_t in the child
> > namespace. Simlarly its process group and session leaders get new pid_ts
> > in the child namespace right ?
> >
> > i.e do the following pid_ts look reasonable when process 1234 unshares
> > its pid namespace ?
> >
> >
> >   PID PPID PGID SID
> >
> > init pid ns 1234 1233 1230 1220
> >
> > child pid ns 3 2 1 0
>
> A slightly more complete answer.
>
> A pid that cannot be represented in the current pid namespace should be
> 0.
>
> pid 1 is very special and in the case of a clone should definitely
> be the first pid in the namespace.
>
> In the case of an unshare pid == 1 is probably the process that does
> the unshare, and it's children all show up in the child namespace.

Sigh, here we go back again to the question of what to do in the case
of a lightweight container which doesn't have a /sbin/init. Let's say
I do
 spawn_container(fork_and_exit(sleep 30m));

so pid 1 forks, pid 2 sleeps 30 minutes, but pid 1 exits right after
the fork. What do we do? Create an idle pid 1? Tack a struct pid
to the parent namespaces' pid=1 making it pid=1 for the child namespace?

-serge

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

Subject: Re: Processes with multiple pid_t values
Posted by ebiederm on Tue, 12 Dec 2006 21:34:20 GMT
View Forum Message <> Reply to Message

"Serge E. Hallyn" <serue@us.ibm.com> writes:

> Sigh, here we go back again to the question of what to do in the case
> of a lightweight container which doesn't have a /sbin/init. Let's say
> I do
> spawn_container(fork_and_exit(sleep 30m));
>
> so pid 1 forks, pid 2 sleeps 30 minutes, but pid 1 exits right after
> the fork. What do we do? Create an idle pid 1? Tack a struct pid
> to the parent namespaces' pid=1 making it pid=1 for the child namespace?

I was thinking whoever called spawn container would be pid 1. But I agree
that is only a partial solution. It definitely makes sense to worry
about this question later when all of the rest of the details for
system containers are sorted out.

Eric

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

Subject: Re: Processes with multiple pid_t values
Posted by Herbert Poetzl on Wed, 13 Dec 2006 08:10:24 GMT
View Forum Message <> Reply to Message

On Tue, Dec 12, 2006 at 02:34:20PM -0700, Eric W. Biederman wrote:
> "Serge E. Hallyn" <serue@us.ibm.com> writes:
>
> > Sigh, here we go back again to the question of what to do in the case
> > of a lightweight container which doesn't have a /sbin/init. Let's say
> > I do
> > spawn_container(fork_and_exit(sleep 30m));
> >
> > so pid 1 forks, pid 2 sleeps 30 minutes, but pid 1 exits right after
> > the fork. What do we do? Create an idle pid 1? Tack a struct pid
> > to the parent namespaces' pid=1 making it pid=1 for the child namespace?

it would be perfectly fine, as long as the following is true:

- only the pid remains, not the associated task
- a 'fallback' reaper takes care of reaping the
  children (in the lightweight case)

- the task creating the container does not need
  to 'hang' around outside the container

> I was thinking whoever called spawn container would be pid 1.
> But I agree that is only a partial solution.

> It definitely makes sense to worry about this question later when all
> of the rest of the details for system containers are sorted out.

okay, as long as 'we' do not forget about this
on the way ...

best,
Herbert

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

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

Subject: Re: Processes with multiple pid_t values
Posted by Sukadev Bhattiprolu on Tue, 09 Jan 2007 03:33:18 GMT
View Forum Message <> Reply to Message

Hmm. I have this mail sitting in my sent folder since Dec 18 but do
not see it in the Containers archives. If you already received, sorry
to spam. Appreciate any clarifications of my doubt :-)

Suka

Date: Mon, 18 Dec 2006 18:37:01 -0800
From: Sukadev Bhattiprolu <sukadev@us.ibm.com>
To: "Eric W. Biederman" <ebiederm@xmission.com>
Cc: Containers <containers@lists.osdl.org>
Subject: Re: Processes with multiple pid_t values

Eric W. Biederman [ebiederm@xmission.com] wrote:
| Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:
|
| > A process that unshares its namespace gets a new pid_t in the child
| > namespace. Simlarly its process group and session leaders get new pid_ts
| > in the child namespace right ?

| >
| > i.e do the following pid_ts look reasonable when process 1234 unshares
| > its pid namespace ?
| >
| >
| >   PID PPID PGID SID
| >
| > init pid ns 1234 1233 1230 1220
| >
| > child pid ns 3 2 1 0
|
| A slightly more complete answer.
|
| A pid that cannot be represented in the current pid namespace should be
| 0.

For my example above, I guess you are saying we want the following ids.

   PID PPID PGID SID

init pid ns 1234 1233 1230 1220

child pid ns 1 0 1 1

If so, my confusion is that for the process 1234, following are true
right ?

 task_pgrp(current) != task_session(current) != task_pid(current)

If we associate a list of [namespace, id] tuples with each struct pid,
then three different struct pids would have the same id in the child
namespace (unless the process calling clone() is both a session and
process group leader).

|
|
| pid 1 is very special and in the case of a clone should definitely
| be the first pid in the namespace.
|
| In the case of an unshare pid == 1 is probably the process that does
| the unshare, and it's children all show up in the child namespace.
|
| Any other pids should simply be allocated with the pid allocator if they
| do not fall into the pid namespace.
|
| > Assuming they are :-), we should expect find_pid() call with nr == 0, 1,
| > or 2 in the child pid namespace to also work right ?

| >
| > But processes 1220, 1230, 1233 are entered into the hash table based on
| > their init pid ns values. And so the above find_pid() calls would not
| > find the process we want.
|
| My expectation is that we will have a link entry that points to the
| real struct pid. Or possibly a different data structure at that point.
| The pid hash table does not scale well to very large numbers of pids.
|
| > i.e some processes have two pid_ts and we want to find them using either
| > of the two values. The pid_hash table can obviously hash on one value.
| >
| > We would need some serious changes to the pid_hash table to do this ???
| > (can we change the hash algorithm to generate a key based on all pid_ts
| > a process has ????)
|
| The key would need to be pid_namespace, pid_t. We just need a few tweaks
| to the lookup algorithm. It's 5-10 line patch most likely.

Yes that makes more sense and works for nested containers.
|
| > Or should we just keep track of these special processes (4 per namespace
| > including the child reaper) in the namespace object - just like we treat
| > the child_reaper special ?
|
| It doesn't look hard to allow pids to appear in several namespaces,
| we need at least one pid to appear in multiple pid namespaces, so
| it makes sense to handle that case. With struct pid and the helper
| functions currently defined it is not hard to allow for all pids to
| be in several namespaces.
|
| So it is my intention that all processes will have a pid in every parent
| pid namespace as well as a pid in their current pid namespace.
|
| Eric

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

Subject: Re: Processes with multiple pid_t values
Posted by ebiederm on Tue, 09 Jan 2007 04:22:10 GMT
View Forum Message <> Reply to Message

Sukadev Bhattiprolu <sukadev@us.ibm.com> writes:

> Hmm. I have this mail sitting in my sent folder since Dec 18 but do

> not see it in the Containers archives. If you already received, sorry
> to spam. Appreciate any clarifications of my doubt :-)

There is something about your usage of the word doubt that does not
quite feel correct. I wonder if it is just me.....

> For my example above, I guess you are saying we want the following ids.
>
>   PID PPID PGID SID
>
> init pid ns 1234 1233 1230 1220
>
> child pid ns 1 0 1 1
>
>
> If so, my confusion is that for the process 1234, following are true
> right ?
>
> task_pgrp(current) != task_session(current) != task_pid(current)
>
> If we associate a list of [namespace, id] tuples with each struct pid,
> then three different struct pids would have the same id in the child
> namespace (unless the process calling clone() is both a session and
> process group leader).

I guess it depends on how we set it up, it all depends on how the groups
are formed.

There is no reason to believe any particular realtionship between
the pid pgrp and session of the calling process from which we
inherit these things. If we can make it all work without doing the
equivalent of setsid() at clone time I think it is a better implementation.

However by default /sbin/init does call setsid() at which point we know
we have the relationship: pid == pgrp == session.

So once setsid() is called (the common case) we will have the id relationship
above. In fact we force that relationship for /sbin/init today at kernel
startup time.

Eric

_____