Subject: Re: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Herbert Poetzl on Fri, 23 Mar 2007 00:57:30 GMT
View Forum Message <> Reply to Message

On Mon, Mar 19, 2007 at 08:04:12PM -0600, Eric W. Biederman wrote:
> Dave Hansen <hansendc@us.ibm.com> writes:
>
> > I was tracking down why we need find_get_pid(1) in
> > proc_get_sb(), when I realized that we apparently
> > don't need a pid at all in the non-pid parts of /proc.
> >
> > Anyone see any problems with this approach?
>
> The thing is these are pid related parts of /proc you are
> working with.
>
>
> I'm trying to remember what the actual semantics were.
>
> I do know doing this means if our pid namespace goes away these
> functions do the right thing.
>
> This may have been how I was getting the pid namespace in originally
> so this code may be obsolete.
>
> Partly I think doing this made the code a little more symmetric.
>
> Regardless I would like to see a little farther down on
> how we test to see if the pid namespace is alive and how we
> make these functions do nothing if it has died. I would also
> like to see how we perform the appropriate lookups by pid namespace.
>
> Basically I want to see how we finish up multiple namespace support
> for /proc before we start with the micro optimizations.
>
>
> I'm fairly certain this patch causes us to do the wrong thing when
> the pid namespace exits, and I don't see much gain except for the
> death of find_get_pid.
>
>
> > For what I would imagine are historical reasons, we set
> > all struct proc_inode->pid fields. We use the init
> > process for all non-/proc/<pid> inodes.
> >
> > We get a handle to the init process in proc_get_sb()
> > then fetch it out in proc_pid_readdir():
> >

> > struct task_struct *reaper =
> > get_proc_task(filp->f_path.dentry->d_inode);
> >
> > The filp in that case is always the root inode on which
> > someone is doing a readdir. This reaper variable gets
> > passed down into proc_base_instantiate() and eventually
> > set in the new inode's ->pid field.
> >
> > The problem is that I don't see anywhere that we
> > actually go and use this, outside of the /proc/<pid>
> > directories. Just referencing the init process like
> > this is a pain for containers because our init process
> > (pid == 1) can actually go away.
>
> Which as far as can recall is part of the point. If you have a pid
> namespace with normal semantics the child reaper pid == 1 is the last
> pid in the pid namespace to exit. Therefore when it exists the pid
> namespace exists and with it doesn't the pid namespace does not exist.

what about lightweight pid spaces, which do not have
a real init process/pid?

IMHO we should define the pid namespace by the
processes and thus it would seize to exist when
the last process leaves the pid space

best,
Herbert

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

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