Subject: Re: [RFC] what the hell is going on with /proc/self?
Posted by ebiederm on Wed, 24 Oct 2007 02:57:26 GMT
View Forum Message <> Reply to Message

Al Viro <viro@ftp.linux.org.uk> writes:

> On Tue, Oct 23, 2007 at 03:20:39PM -0500, Matt Mackall wrote:
>> On Tue, Oct 23, 2007 at 03:03:36AM +0100, Al Viro wrote:
>> >  What is the proc_base_stuff[] nonsense about?  AFAICS, that
>> > went in with no reason whatsoever in
>> > commit 801199ce805a2412bbcd9bfe213092ec656013dd
>> > Author: Eric W. Biederman <ebiederm@xmission.com>
>> > Date:   Mon Oct 2 02:18:48 2006 -0700
>> >
>> >  Rationale is very weak and patch adds considerable complexity
>> > for no good reason.  Besides, it's obfuscated just for the hell of it:
>> >  if (!IS_ERR(result) || PTR_ERR(result) != -ENOENT)
>> > instead of
>> >  if (result != ERR_PTR(-ENOENT))
>> > etc.
>> >
>> >  Unless there are _real_ plans that would justify that animal,
>> > I'm going to get rid of it in the pending patch series (/proc/self
>> > cleanups, saner dentry retention for non-process parts, etc.).
>>
>> Seems obvious to cc: Eric.
>
> Doh...  Sorry, thought I'd done that.  Eric, my apologies.

No problem.

It has been a while so let me see if I can dredge up what goes
on there, partly I ran out of steam when working on that.

One useful aspect of that change to use common infrastructure was in
removing the hard coded inode numbers from /proc.  By going through
proc_fill_cache  I was able to ensure the inode numbers matched
up for /proc/self no matter what they were.

Another aspect of the change that I didn't feel comfortable using
to justify it then and but I do now think is important now is that if
the pid namespace goes away (aka a secondary init and all their
children exit) /proc/self disappears.

I believe that my original patch was smaller and had a bunch more
code reuse until I discovered that the I would goof up the
security modules if I called security_task_to_inode on /proc/self.

So it is my desire (and I think it a reasonable one) that if all
of the tasks in a pid namespace that correspond to a mount of proc
exit then all of the files associated with the pid namespace itself
should disappear.

Getting all of the files that are process related into the
infrastructure of fs/proc/base.c is one way to achieve the process
related files disappear. Especially as it seemed well connected with
something the concept of splitting proc up into it's constituent
filesystems. /proc/<pid> /proc/sys and /proc/{generic}.

The more I look at that the amount of /proc that doesn't become
namespace related and thus desirable to be show per process is getting
quite small, so a different technique then using the infrastructure
in fs/proc/base.c may be desirable.

When eventually we get to the device namespace (bleh) we get
things like /proc/scsi/ /proc/tty/ /proc/ide/ and /proc/devices
that should become per namespace as well. Which probably means
at least half of the existing of the existing /proc/{generic}
stuff looks to become per namespace.

So in practice the things that I see needing to happen with /proc
right now.
- Figure out how to move /proc/net into /proc/<pid>/net leaving
  behind a /proc/net symlink to /proc/self/net.
- Fix proc_kill_inodes to cope with multiple super blocks.
- /proc/self I still think needs to get the pid and not the tgid
  as sometimes I think we get incorrect behavior when coming from
  a thread.
- /proc/sysvipc should really become a /proc/self/sysvipc symlink.
- /proc/sys needs to become /proc/self/sys/ and the work pushed
  down in that direction.

For the stuff that isn't per namespace enhancing the caching does
have the challenge that we are likely to hit the current bug in
proc_kill_inodes because the inodes will stay around longer.
Although ideally if this could be it's own filesystem we would
only have a single instance of those dentries.

Similarly it would be nice if per namespace things like
/proc/<pid>/mounts, /proc/<pid>/net, /proc/<pid>/sys,
/proc/<pid>/sysvipc could share the same dentry trees,
across different /proc/<pid> instances. Especially as that
would allow using stat to detect if two processes were sharing
the same namespace. The only thing that suggests itself is making
kernel mounts that are per namespace, for these things. Which is one
of the reasons I'm interested in splitting /proc up into separate

filesystems.

So in summary.  I really don't care how the internals of /proc look, or which path we take.  As long as we do improve /proc and ultimately sort through the per namespace details.

Eric

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