Subject: Re: [PATCH v3] SUNRPC: set desired file system root before connecting local transports
Posted by ebiederm on Wed, 10 Oct 2012 02:00:35 GMT
View Forum Message <> Reply to Message

ebiederm@xmission.com (Eric W. Biederman) writes:

> "J. Bruce Fields" <bfields@fieldses.org> writes:
>
>> On Tue, Oct 09, 2012 at 01:20:48PM -0700, Eric W. Biederman wrote:
>>> "Myklebust, Trond" <Trond.Myklebust@netapp.com> writes:
>>>
>>> > On Tue, 2012-10-09 at 15:35 -0400, J. Bruce Fields wrote:
>>> >> Cc'ing Eric since I seem to recall he suggested doing it this way?
>>>
>>> Yes. On second look setting fs->root won't work. We need to change fs.
>>> The problem is that by default all kernel threads share fs so changing
>>> fs->root will have non-local consequences.
>>
>> Oh, huh. And we can't "unshare" it somehow?
>
> I don't fully understand how nfs uses kernel threads and work queues.
> My general understanding is work queues reuse their kernel threads
> between different users. So it is mostly a don't pollute your
> environment thing. If there was a dedicated kernel thread for each
> environment this would be trivial.
>
> What I was suggesting here is changing task->fs instead of
> task->fs.root. That should just require task_lock().
>
>> Or, previously you suggested:
>>
>> - introduce sockaddr_fd that can be applied to AF_UNIX sockets,
>>   and teach unix_bind and unix_connect how to deal with a second
>>   type of sockaddr, AT_FD:
>>   struct sockaddr_fd { short fd_family; short pad; int fd; }
>>
>> - introduce sockaddr_unix_at that takes a directory file
>>   descriptor as well as a unix path, and teach unix_bind and
>>   unix_connect to deal with a second sockaddr type, AF_UNIX_AT:
>>   struct sockaddr_unix_at { short family; short pad; int dfd; char path[102]; }
>>
>> Any other options?
>
> I am still half hoping we don't have to change the userspace API/ABI.
> There is sanity checking on that path that no one seems interested in to
> solve this problem.

There is a good option if we are up to userspace ABI extensions.

Implement open(2) on unix domain sockets. Where open(2) would
essentially equal connect(2) on unix domain sockets.

With an open(2) implementation we could use file_open_path and the
implementation should be pretty straight forward and maintainable.
So implementing open(2) looks like a good alternative implementation
route.

Eric
