Subject: Re: [RFC PATCH 0/5] Resend - Use procfs to change a syscall behavior
Posted by Nadia Derbey on Fri, 04 Jul 2008 12:07:18 GMT
View Forum Message <> Reply to Message

Pavel Machek wrote:
> Hi!
>
>
>>This patchset is a part of an effort to change some syscalls behavior for
>>checkpoint restart.
>>
>>When restarting an object that has previously been checkpointed, its state
>>should be unchanged compared to the checkpointed image.
>>For example, a restarted process should have the same upid nr as the one it
>>used to have when being checkpointed; an ipc object should have the same id
>>as the one it had when the checkpoint occured.
>>Also, talking about system V ipcs, they should be restored with the same
>>state (e.g. in terms of pid of last operation).
>>
>>This means that several syscalls should not behave in a default mode when
>>they are called during a restart phase.
>>
>>One solution consists in defining a new syscall for each syscall that is
>>called during restart:
>> . sys_fork_with_id() would fork a process with a predefined id.
>> . sys_msgget_with_id() would create a msg queue with a predefined id
>> . sys_semget_with_id() would create a semaphore set with a predefined id
>> . etc,
>>
>>This solution requires defining a new syscall each time we need an existing
>>syscall to behave in a non-default way.
>
>
> Yes, and I believe that's better than...
>
>
>>An alternative to this solution consists in defining a new field in the
>>task structure (let's call it next_syscall_data) that, if set, would change
>>the behavior of next syscall to be called. The sys_fork_with_id() previously
>>cited can be replaced by
>> 1) set next_syscall_data to a target upid nr
>> 2) call fork().
>
>
> ...bloat task struct and
>
>
>>A new file is created in procfs: /proc/self/task/<my_tid>/next_syscall_data.

>>This makes it possible to avoid races between several threads belonging to
>>the same process.
>
>
> ...introducing this kind of uglyness.
>
> Actually, there were proposals for sys_indirect(), which is slightly
> less ugly, but IIRC we ended up with adding syscalls, too.
> Pavel

Pavel,

I had a look at the lwn.net article that describes the sys_indirect()
interface.
It does exactly what we need here, so I do like it, but it has the same
drawbacks as the one you're complaining about:
. a new field is needed in the task structure
. looks like many people found it ugly...

Now, coming back to what I'm proposing: what we need is actually to
change the behavior of *existing* syscalls, since we are in a very
particular context (restarting an application).

Defining brand new syscalls is very touchy: needs to be careful about
the interface + I can't imagine the number of syscalls that would be needed.

Now, since we do have a set of available syscalls, I think it's much
easier to change their behavior depending on a field being set in the
task structure.

I agree with you that the interface is not that nice, so what about
proposing a single syscall that would set the next_syscall_data field in
the task structure (instead of setting it through procfs).
It's true that this makes us end up with a "2 passes" sys_indirect()
(i.e. 2 syscalls called instead of a single one), but it is much
simpler. And may be the induced performance overhead would not be that
important since we are, again, in a particular context (restarting an
application)?

Regards,
Nadia