Subject: Re: [RFC][PATCH 0/4] Object creation with a specified id
Posted by Oren Laadan on Fri, 14 Mar 2008 15:50:13 GMT
View Forum Message <> Reply to Message

Nadia Derbey wrote:
> Oren Laadan wrote:
>>
>>
>> Nadia.Derbey@bull.net wrote:
>>
>>> A couple of weeks ago, a discussion has started after Pierre's
>>> proposal for
>>> a new syscall to change an ipc id (see thread
>>> http://lkml.org/lkml/2008/1/29/209).
>>>
>>>
>>> Oren's suggestion was to force an object's id during its creation,
>>> rather
>>> than 1. create it, 2. change its id.
>>>
>>> So here is an implementation of what Oren has suggested.
>>>
>>> 2 new files are defined under /proc/self:
>>>   . next_ipcid --> next id to use for ipc object creation
>>>   . next_pids --> next upid nr(s) to use for next task to be forked
>>>   (see patch #2 for more details).
>>
>>
>> Generally looks good. One meta-comment, though:
>>
>> I wonder why you use separate files for separate resources,
>
> That would be needed in a situation wheere we don't care about next,
> say, ipc id to be created but we need a predefined pid. But I must admit
> I don't see any pratical application to it.

exactly; why set the next-ipc value so far in advance ? I think it's
better (and less confusing) if we require that setting the next-id value
be done right before the respective syscall.

>
>> and why you'd
>> want to write multiple identifiers in one go;
>
> I used multiple identifiers only for the pid values: this is because
> when a new pid value is allocated for a process that belongs to nested
> namespaces, the lower level upid nr values are allocated in a single
> shot. (see alloc_pid()).

>
>> it seems to complicate the
>> code and interface with minimal gain.
>> In practice, a process will only do either one or the other, so a single
>> file is enough (e.g. "next_id").
>> Also, writing a single value at a time followed by the syscall is enough;
>> it's definitely not a performance issue to have multiple calls.
>> We assume the user/caller knows what she's doing, so no need to classify
>> the identifier (that is, tell the kernel it's a pid, or an ipc id) ahead
>> of time. The caller simply writes a value and then calls the relevant
>> syscall, or otherwise the results may not be what she expected...
>> If such context is expected to be required (although I don't see any at
>> the moment), we can require that the user write "TYPE VALUE" pair to
>> the "next_id" file.
>
> That's exactly what I wanted to avoid by creating 1 file per object.
> Now, it's true that in a restart context where I guess that things will
> be done synchronously, we could have a single next_id file.
>
>>
>>>
>>> When one of these files (or both of them) is filled, a structure
>>> pointed to
>>> by the calling task struct is filled with these ids.
>>>
>>> Then, when the object is created, the id(s) present in that structure
>>> are
>>> used, instead of the default ones.
>>>
>>> The patches are against 2.6.25-rc3-mm1, in the following order:
>>>
>>> [PATCH 1/4] adds the procfs facility for next ipc to be created.
>>> [PATCH 2/4] adds the procfs facility for next task to be forked.
>>> [PATCH 3/4] makes use of the specified id (if any) to allocate the
>>> new IPC
>>>   object (changes the ipc_addid() path).
>>> [PATCH 4/4] uses the specified id(s) (if any) to set the upid nr(s)
>>> for a newly
>>>   allocated process (changes the alloc_pid()/alloc_pidmap()
>>> paths).
>>>
>>> Any comment and/or suggestions are welcome.
>>>
>>> Cc-ing Pavel and Sukadev, since they are the pid namespace authors.
>>>
>>> Regards,
>>> Nadia
>>>

>>> --
>>>
>>> --
>>
>>
>>
>
>
> Regards,
> Nadia