Subject: Re: Namespaces exhausted CLONE_XXX bits problem
Posted by Cedric Le Goater on Mon, 14 Jan 2008 15:20:19 GMT
View Forum Message <> Reply to Message

>>> I started looking at PTYs/TTYs/Console to make the appropriate
>>> namespace and suddenly remembered that we have already
>>> exhausted all the CLONE_ bits in 32-bit mask.
>> yes nearly. 1 left with the mq_namespace i'm going to send.
>
> Yup. That's why I think that we should first solve this
> issue and then send more namespaces.

OK.

>>> So, I recalled the discussions we had and saw the following
>>> proposals of how to track this problem (with their disadvantages):
>>>
>>> 1. make the clone2 system call with 64-bit mask
>>>   - this is a new system call
>> sys_clone2 is used on ia64 ... so we would need another name.
>>
>> clone_ns() would be nice but it's too specific to namespaces unless
>> we agree that we need a new syscall specific to namespaces.
>>
>> clone_new or clone_large ?
>
> clone3 :) Just kidding. _If_ implement new system calls then I'd
> better like cloe_ns and unshare_nr pair.

We will find a name.

>>> 2. re-use CLONE_STOPPED
>>>   - this will give us only one bit
>> not enough.
>
> Yup :)
>
>>> 3. merge existing bits into one
>>>   - we lose the ability to create them separately
>> it would be useful to have such a flag though, something like CLONE_ALLN,
>> because it's the one everyone is going to use.
>>
>> what i've been looking at in December is 1. and 3. : a new general purpose
>> clone syscall with extend flags. The all-in-on flag is not an issue but it
>> would be nice to keep the last clone flag for this purpose.
>>
>> Now, if we use 64bits, we have a few issue/cleanups to solve. First, in
>> kernel land, the clone_flags are passed down to the security modules

```
>>
>>  security_task_create()
>>
>> so we'll have to change to kernel api. I don't remember anything else
>> blocking.
>>
>> In user land, we need to choose a prototype supporting also 32bits arches.
>> so it could be :
>>
>>  long sys_clone_new(struct clone_new_args)
>>
>> or
>>
>>  long sys_clone_new(... unsigned long flags_high, unsigned long flag_low ...)
>>
>> Second option might be an issue because clone already has 6 arguments.
>> right ?
>
> Yes.
>
>>> 4. implement a sys_unshare_ns system call with 64bit/arbitrary mask
>>>    - this is anew system call
>> I think that a new clone deserves a new unshare.
>>
>>>    - this will bring some dissymmetry between namespaces
>> what do you mean ?
>
> I mean, that soe namespaces will be unshare-only, but some
> clone-and-unshare.

OK. we still have that already. pid namespace for instance.

>>> 5. use sys_indirect
>>>    - this one is not in even -mm tree yet and it's questionable
>>>      whether it will be at all
>> I don't know much about that one.
>>
>> C.
>
> So you seem to prefer a "new system call" approach, right?

yes.

to be more precise :

 long sys_clone_something(struct clone_something_args args)

and
```

long sys_unshare_something(struct unshare_something_args args)

The arg passing will be slower bc of the copy_from_user() but we will
still have the sys_clone syscall for the fast path.

C.