Subject: [RFC PATCH 0/5] Resend -v2 - Use procfs to change a syscall behavior
Posted by Nadia Derbey on Tue, 08 Jul 2008 11:24:22 GMT
View Forum Message <> Reply to Message

Resending after fixing the issues pointed out by Serge.

Also ported to 2.6.26-rc8-mm1.

Regards,
Nadia

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.

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().

This patch series implements the 2nd solution. Actually I've already sent it some times ago, and things ended up with Pavel complaining about the "ugly interface" (see
https://lists.linux-foundation.org/pipermail/containers/2008-April/010909.html).

Now, I'm resending the series because this 2nd solution has the advantage of being easily reusable for many subsystems: the only thing needed is just to set a field in the task structure and rewrite the code portion that is sensitive to this field being set (it's successfully being used in cryo code - git tree at git://git.sr71.net/~hallyn/cryodev.git).

The patches have been ported to 2.6.26-rc8-mm1 and the open() syscall in now covered.

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.

Setting a value into this file fills in the next_syscall_data in the task structure.

The following subsystems have been changed to take this value into account:
1) sysvipc:
   . if there's a value in next_syscall_data when msgget() is called, msgget()
     creates a msg queue with that value as an id
   . this applies to semget() and shmget().
   . if next_syscall_data is set to 1 when msgctl(IPC_SET) is called, msgctl()
     sets more that the usual permission fields for the target msg queue (it
     sets the time fields, and the pid of last operation fields).
   . this applies to semctl() and shmctl().
2) process creation:
   . if there's a value in next_syscall_data when fork() is called, fork()
     creates a process with that value as a pid.
   . this applies to vfork() and clone().
3) file descriptors:
   . if there's a value in next_syscall_data when open() is called, open()
     uses that value as the file descriptor for the open file

The syntax is:
```
# echo "LONG1 XX" > /proc/self/task/<my_tid>/next_syscall_data
    next object to be created will have an id set to XX
```

Today, the ids are specified as long, but having a type string specified in the next_syscall_data file makes it possible to cover more types in the future, if needed.
Also, only a single value can be set. But the number that immediatly follows the type string makes it possible to specify more values in the future, if needed. This can be applied, e.g. to predefine all the upid nrs for a process that belongs to nested namespaces, if needed in the future.

These patches should be applied to 2.6.26-rc8-mm1, in the following order:

[PATCH 1/5] : next_syscall_data_proc_file.patch
[PATCH 2/5] : ipccreate_use_next_syscall_data.patch
[PATCH 3/5] : proccreate_use_next_syscall_data.patch
[PATCH 4/5] : ipcset_use_next_syscall_data.patch
[PATCH 5/5] : fileopen_use_next_syscall_data.patch

Any comment and/or suggestions are welcome.

Regards,
Nadia

--

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