
Subject: Re: [PATCH] An attempt to have an unlimitedly extendable sys_clone
Posted by [serue](#) on Tue, 15 Jan 2008 21:40:18 GMT

[View Forum Message](#) <> [Reply to Message](#)

Quoting Cedric Le Goater (clg@fr.ibm.com):

> Cedric Le Goater wrote:

> > Pavel Emelyanov wrote:

> > > We have one bit in the clone_flags left, so we won't be
> > > able to create more namespaces after we make it busy.
> > > Besides, for checkpoint/restart jobs we might want to
> > > create tasks with pre-defined pids (virtual of course).
> > > What else might be required from clone() - nobody knows.

> > >

> > > This is an attempt to create a extendable API for clone.

> > >

> > > I use the last bit in the clone_flags for CLONE_NEWCLONE.

> > > When set it will denote that the child_tidptr is not a
> > > pointer on the tid storage, but the pointer on the struct
> > > long_clone_struct which currently looks like this:

> > >

> > > struct long_clone_arg {

> > > int size;

> > > };

> > >

> > > When we want to add a new argument for clone we just put
> > > it *at the end of this structure* and adjust the size.

> > > The binary compatibility with older long_clone_arg-s is
> > > facilitated with the clone_arg_has() macro.

> > >

> > > hmm, I wonder how lkml@ will react to this. do we have
> > > similar apis in the kernel ?

> > >

> > > Sure, we lose the ability to clone tasks with extended
> > > argument and the CLONE_CHILD_SETTID/CLEARTID, but do we
> > > really need this?

> > >

> > > not in the extended clone flag version. I think.

> > >

> > > The same thing is about to be done for unshare - we can
> > > add the second argument for it and iff the CLONE_NEWCLONE
> > > is specified - try to use it. Binary compatibility with
> > > the old unshare will be kept.

> > >

> > > The new argument is pulled up to the create_new_namespaces
> > > so that later we can easily use it w/o sending additional
> > > patches.

> > >

> > > This is a final, but a pre-review patch for sys_clone()

```

> >> that I plan to send to Andrew before we go on developing
> >> new namespaces.
> >>
> >> Made against 2.6.24-rc5-mm1.
> >
> > The patch looks good and I compiled it and booted on x64 and
> > x86_64.
> >
> > I think we should add the unshare support before sending to
> > andrew and also add an extended flag array to show how it will
> > be used. I have a mq_namespace patchset pending we could use
> > for that and send all together ?
>
> Here's the unshare part if you want to fold that with your patch.
>
> Thanks,
>
> C.
>
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> ---
> include/linux/nsproxy.h | 2 +-
> include/linux/syscalls.h | 2 +-
> kernel/fork.c           | 19 ++++++++-----
> kernel/nsproxy.c        | 7 +++++---
> 4 files changed, 21 insertions(+), 9 deletions(-)
>
> Index: 2.6.24-rc5-mm1/include/linux/nsproxy.h
> =====
> --- 2.6.24-rc5-mm1.orig/include/linux/nsproxy.h
> +++ 2.6.24-rc5-mm1/include/linux/nsproxy.h
> @@ -68,7 +68,7 @@ void exit_task_namespaces(struct task_st
> void switch_task_namespaces(struct task_struct *tsk, struct nsproxy *new);
> void free_nsproxy(struct nsproxy *ns);
> int unshare_nsproxy_namespaces(unsigned long, struct nsproxy **,
> - struct fs_struct *);
> + struct fs_struct *, struct long_clone_arg *carg);
>
> static inline void put_nsproxy(struct nsproxy *ns)
> {
> Index: 2.6.24-rc5-mm1/include/linux/syscalls.h
> =====
> --- 2.6.24-rc5-mm1.orig/include/linux/syscalls.h
> +++ 2.6.24-rc5-mm1/include/linux/syscalls.h
> @@ -585,7 +585,7 @@ asmlinkage long compat_sys_newfstatat(un
> int flag);
> asmlinkage long compat_sys_openat(unsigned int dfd, const char __user *filename,
> int flags, int mode);

```

```
> -asmlinkage long sys_unshare(unsigned long unshare_flags);
> +asmlinkage long sys_unshare(unsigned long unshare_flags, int __user *flagptr);
```

Hmm, why not properly call this a struct long_clone_arg __user *flagptr here?

```
>
> asmlinkage long sys_splice(int fd_in, loff_t __user *off_in,
>                             int fd_out, loff_t __user *off_out,
> Index: 2.6.24-rc5-mm1/kernel/fork.c
> =====
> --- 2.6.24-rc5-mm1.orig/kernel/fork.c
> +++ 2.6.24-rc5-mm1/kernel/fork.c
> @@ -1700,7 +1700,7 @@ static int unshare_semundo(unsigned long
>  * constructed. Here we are modifying the current, active,
>  * task_struct.
>  */
> -asmlinkage long sys_unshare(unsigned long unshare_flags)
> +asmlinkage long sys_unshare(unsigned long unshare_flags, int __user *flagptr)
> {
>     int err = 0;
>     struct fs_struct *fs, *new_fs = NULL;
> @@ -1709,6 +1709,7 @@ asmlinkage long sys_unshare(unsigned lon
>     struct files_struct *fd, *new_fd = NULL;
>     struct sem_undo_list *new_ulist = NULL;
>     struct nsproxy *new_nsproxy = NULL;
> +     struct long_clone_arg *carg = NULL;
>
>     check_unshare_flags(&unshare_flags);
>
> @@ -1717,11 +1718,19 @@ asmlinkage long sys_unshare(unsigned lon
>     if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
>                             CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
>                             CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
> -                             CLONE_NEWNET))
> +                             CLONE_NEWNET|CLONE_NEWCLONE))
>         goto bad_unshare_out;
>
> +     if (unshare_flags & CLONE_NEWCLONE) {
> +         carg = get_long_clone_arg(flagptr);
> +         if (IS_ERR(carg)) {
> +             err = PTR_ERR(carg);
> +             goto bad_unshare_out;
> +         }
> +     }
>
>     if ((err = unshare_thread(unshare_flags)))
> -         goto bad_unshare_out;
```

```

> +         goto bad_unshare_cleanup_carg;
>     if ((err = unshare_fs(unshare_flags, &new_fs)))
>         goto bad_unshare_cleanup_thread;
>     if ((err = unshare_sighand(unshare_flags, &new_sigh)))
> @@ -1733,7 +1742,7 @@ asmlinkage long sys_unshare(unsigned lon
>     if ((err = unshare_semundo(unshare_flags, &new_ulist)))
>         goto bad_unshare_cleanup_fd;
>     if ((err = unshare_nsproxy_namespaces(unshare_flags, &new_nsproxy,
> -         new_fs)))
> +         new_fs, carg)))
>         goto bad_unshare_cleanup_semundo;
>
>     if (new_fs || new_mm || new_fd || new_ulist || new_nsproxy) {
> @@ -1791,6 +1800,8 @@ bad_unshare_cleanup_fs:
>         put_fs_struct(new_fs);
>
> bad_unshare_cleanup_thread:
> +bad_unshare_cleanup_carg:
> +    kfree(carg);
> bad_unshare_out:
>     return err;
> }
> Index: 2.6.24-rc5-mm1/kernel/nsproxy.c
> =====
> --- 2.6.24-rc5-mm1.orig/kernel/nsproxy.c
> +++ 2.6.24-rc5-mm1/kernel/nsproxy.c
> @@ -182,19 +182,20 @@ void free_nsproxy(struct nsproxy *ns)
>  * On success, returns the new nsproxy.
>  */
> int unshare_nsproxy_namespaces(unsigned long unshare_flags,
> -     struct nsproxy **new_nsp, struct fs_struct *new_fs)
> +     struct nsproxy **new_nsp, struct fs_struct *new_fs,
> +     struct long_clone_arg *carg)
> {
>     int err = 0;
>
>     if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
> -         CLONE_NEWUSER | CLONE_NEWNET)))
> +         CLONE_NEWUSER | CLONE_NEWNET | CLONE_NEWCLONE)))
>         return 0;
>
>     if (!capable(CAP_SYS_ADMIN))
>         return -EPERM;
>
>     *new_nsp = create_new_namespaces(unshare_flags, current,
> -         new_fs ? new_fs : current->fs, NULL);
> +         new_fs ? new_fs : current->fs, carg);
>     if (IS_ERR(*new_nsp)) {

```

```
>         err = PTR_ERR(*new_nsp);  
>         goto out;
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
