
Subject: Re: [RFC][PATCH 6/6]: Enable unsharing pid namespace.

Posted by [ebiederm](#) on Sun, 11 Mar 2007 12:01:51 GMT

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

sukadev@us.ibm.com writes:

```
> From: Cedric Le Goater <clg@fr.ibm.com>
> Subject: [RFC][PATCH 6/6]: Enable unsharing pid namespace.
>
> Enable unsharing of pid namespace - i.e allow creating and using
> a new pid namespace and attaching multiple struct pid_nrs per
> process, one for each namespace.
```

```
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
```

```
>
> ---
> include/linux/sched.h | 1 +
> kernel/fork.c          | 22 ++++++++-----
> kernel/nsproxy.c       | 5 ++++
> kernel/pid.c           | 18 ++++++++
> 4 files changed, 41 insertions(+), 5 deletions(-)
```

```
> Index: lx26-20-mm2b/kernel/pid.c
```

```
> =====
> --- lx26-20-mm2b.orig/kernel/pid.c 2007-03-09 14:56:53.000000000 -0800
> +++ lx26-20-mm2b/kernel/pid.c 2007-03-09 15:03:05.000000000 -0800
> @@ -546,6 +546,24 @@ void free_pid_ns(struct kref *kref)
> struct pid_namespace *ns;
>
> ns = container_of(kref, struct pid_namespace, kref);
> +
> + if (ns != &init_pid_ns) {
```

A cleaned up version of this would make sense.

We don't need to test that ns != init_pid_ns.

Maybe BUG_ON(ns == &init_pid_ns);

Or do you foresee freeing the initial pid namespace?

```
> + /* BEGIN: to be removed soon */
> + int i;
> + int nr_free = atomic_read(&ns->pidmap[0].nr_free);
> +
> + BUG_ON(nr_free != (BITS_PER_PAGE - 1));
> +
> + if (unlikely(PIDMAP_ENTRIES != 1)) {
```

```
> + for (i = 1; i < PIDMAP_ENTRIES; i++) {
> +   nr_free = atomic_read(&ns->pidmap[i].nr_free);
> +   BUG_ON(nr_free != BITS_PER_PAGE);
> + }
> + }
```

What makes the first page in the pidmap special? None of the pages should have an allocated bit if we are freeing the pid namespace....

```
> + /* END: to be removed soon */
> + kfree(ns->pidmap[0].page);
```

This is wrong. While a pid namespace exists we don't free the page so you need to loop through all of the pidmap entries and free them.

```
> + }
> +
>   kfree(ns);
> }
>
> Index: lx26-20-mm2b/include/linux/sched.h
> =====
> --- lx26-20-mm2b.orig/include/linux/sched.h 2007-03-09 14:56:12.000000000 -0800
> +++ lx26-20-mm2b/include/linux/sched.h 2007-03-09 14:56:53.000000000 -0800
> @@ -26,6 +26,7 @@
> #define CLONE_STOPPED 0x02000000 /* Start in stopped state */
> #define CLONE_NEWUTS 0x04000000 /* New utsname group? */
> #define CLONE_NEWIPC 0x08000000 /* New ipc */
> +#define CLONE_NEWPID 0x10000000 /* New pid namespace */
```

Good. Although it may make sense to submit this as a separate patch. We should be able to reserve the number now.

```
> Index: lx26-20-mm2b/kernel/fork.c
> =====
> --- lx26-20-mm2b.orig/kernel/fork.c 2007-03-09 14:56:12.000000000 -0800
> +++ lx26-20-mm2b/kernel/fork.c 2007-03-09 15:05:39.000000000 -0800
> @@ -57,6 +57,7 @@
> #include <asm/mmu_context.h>
> #include <asm/cacheflush.h>
> #include <asm/tlbflush.h>
> +#include <linux/pid_namespace.h>
```

I don't see `do_fork` support only `unshare` support which seems completely backwards to me.

```
> @@ -1609,6 +1610,7 @@ asmlinkage long sys_unshare(unsigned lon
>   struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
```

```

> struct uts_namespace *uts, *new_uts = NULL;
> struct ipc_namespace *ipc, *new_ipc = NULL;
> + struct pid_nr *new_pid_nr = NULL;
>
> check_unshare_flags(&unshare_flags);
>
> @@ -1616,7 +1618,7 @@ asmlinkage long sys_unshare(unsigned lon
> err = -EINVAL;
> if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
> CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
> - CLONE_NEWUTS|CLONE_NEWIPC))
> + CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWPID))
> goto bad_unshare_out;
>
> if ((err = unshare_thread(unshare_flags)))
> @@ -1637,18 +1639,20 @@ asmlinkage long sys_unshare(unsigned lon
> goto bad_unshare_cleanup_semundo;
> if ((err = unshare_ipcs(unshare_flags, &new_ipc)))
> goto bad_unshare_cleanup_uts;
> + if ((err = unshare_pid_ns(unshare_flags, &new_pid_nr)))
> + goto bad_unshare_cleanup_ipc;
>
> - if (new_ns || new_uts || new_ipc) {
> + if (new_ns || new_uts || new_ipc || new_pid_nr) {
> old_nsproxy = current->nsproxy;
> new_nsproxy = dup_namespaces(old_nsproxy);
> if (!new_nsproxy) {
> err = -ENOMEM;
> - goto bad_unshare_cleanup_ipc;
> + goto bad_unshare_cleanup_pid;
> }
> }
>
> if (new_fs || new_ns || new_mm || new_fd || new_ulist ||
> - new_uts || new_ipc) {
> + new_uts || new_ipc || new_pid_nr) {
>
> task_lock(current);
>
> @@ -1696,12 +1700,22 @@ asmlinkage long sys_unshare(unsigned lon
> new_ipc = ipc;
> }
>
> + if (new_pid_nr) {
> + pid = task_pid_ns(current);
> + set_task_pid_ns(current, new_pid_nr->pid_ns);
> + new_pid_nr = NULL;
> + }

```

```

> +
> task_unlock(current);
> }
>
> if (new_nsproxy)
> put_nsproxy(new_nsproxy);
>
> +bad_unshare_cleanup_pid:
> + if (new_pid_nr)
> + free_pidmap_pid_nr(new_pid_nr);
> +
> bad_unshare_cleanup_ipc:
> if (new_ipc)
> put_ipc_ns(new_ipc);
> Index: lx26-20-mm2b/kernel/nsproxy.c
> =====
> --- lx26-20-mm2b.orig/kernel/nsproxy.c 2007-03-09 14:56:12.000000000 -0800
> +++ lx26-20-mm2b/kernel/nsproxy.c 2007-03-09 15:03:05.000000000 -0800
> @@ -83,13 +83,16 @@ int copy_namespaces(int flags, struct ta
> struct nsproxy *old_ns = tsk->nsproxy;
> struct nsproxy *new_ns;
> int err = 0;
> + int ns_all;
>
> if (!old_ns)
> return 0;
>
> get_nsproxy(old_ns);
>
> - if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
> + ns_all = CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWPID;
> +

```

This doesn't quite seem to make sense why the extra intermediate variable?

```

> + if (!(flags & ns_all))
> return 0;
>

```

Eric

Containers mailing list
Containers@lists.osdl.org
<https://lists.osdl.org/mailman/listinfo/containers>
