Subject: Re: [PATCH 2/2] extend clone_flags using parent_tidptr argument
Posted by serue on Mon, 04 Feb 2008 20:24:16 GMT
View Forum Message <> Reply to Message

Quoting Cedric Le Goater (legoater@free.fr):
> From: Cedric Le Goater <clg@fr.ibm.com>
>
> We have at least 2 patchsets requiring each a new clone flag and there it
> is, we've reached the limit, none are left.
> This patch uses the CLONE_DETACHED flag (unused) as a marker to extend the

Are we pretty sure that there is no legacy software out there which has
continued to specify CLONE_DETACHED since the kernel ignores it?

> clone flags through the parent_tidptr argument.
>
> Initially, we thought on using the last bit but it has recently been taken
> by CLONE_IO.
>
> Obviously, this hack doesn't work for unshare() for which I don't see any
> other solution than to add a new syscall :
> long sys_unshare64(unsigned long clone_flags_high,   unsigned long
> clone_flags_low);
>
>
>
> Is this the right path to extend the clone flags ? should we add a
> clone64() rather than hack the extending clone() ?
> Thanks for any comments !
>
> C.
>
> Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
> ---
> include/linux/sched.h | 1 +
> kernel/fork.c | 14 ++++++++++++-
> 2 files changed, 14 insertions(+), 1 deletion(-)
>
> Index: 2.6.24-mm1/include/linux/sched.h
> ===
> --- 2.6.24-mm1.orig/include/linux/sched.h
> +++ 2.6.24-mm1/include/linux/sched.h
> @@ -28,6 +28,7 @@
> #define CLONE_NEWPID 0x20000000 /* New pid namespace */
> #define CLONE_NEWNET 0x40000000 /* New network namespace */
> #define CLONE_IO 0x80000000 /* Clone io context */
> +#define CLONE_EXTFLAGS CLONE_DETACHED /* use parent_tidptr as an extended
> set of flags */

>
> /*
> * Scheduling policies
> Index: 2.6.24-mm1/kernel/fork.c
> ===
> --- 2.6.24-mm1.orig/kernel/fork.c
> +++ 2.6.24-mm1/kernel/fork.c
> @@ -1012,6 +1012,14 @@ static struct task_struct *copy_process(
> struct task_struct *p;
> int cgroup_callbacks_done = 0;
>
> + /*
> + * It is not permitted to specify both CLONE_EXTFLAGS and
> + * CLONE_PARENT_SETTID
> + */
> + if ((clone_flags & (CLONE_EXTFLAGS|CLONE_PARENT_SETTID)) ==
> +   (CLONE_EXTFLAGS|CLONE_PARENT_SETTID))
> + return ERR_PTR(-EINVAL);
> +
> if ((clone_flags & (CLONE_NEWNS|CLONE_FS)) == (CLONE_NEWNS|CLONE_FS))
>   return ERR_PTR(-EINVAL);
>
> @@ -1455,6 +1463,7 @@ long do_fork(unsigned long clone_flags,
> struct task_struct *p;
> int trace = 0;
> long nr;
> + u64 clone_flags64 = clone_flags;
>
> /*
>   * We hope to recycle these flags after 2.6.26
> @@ -1479,7 +1488,10 @@ long do_fork(unsigned long clone_flags,
>   clone_flags |= CLONE_PTRACE;
> }
>
> - p = copy_process(clone_flags, stack_start, regs, stack_size,
> + if (clone_flags & CLONE_EXTFLAGS)
> + clone_flags64 = ((u64) (uintptr_t) parent_tidptr << 32) | clone_flags;
> +
> + p = copy_process(clone_flags64, stack_start, regs, stack_size,
>   child_tidptr, NULL);
> /*
>   * Do this prior waking up the new thread - the thread pointer
> _____
> Containers mailing list
> Containers@lists.linux-foundation.org
> https://lists.linux-foundation.org/mailman/listinfo/containers

_____
Containers mailing list