
Subject: Re: [patch -mm 10/17] nsproxy: add unshare_ns and bind_ns syscalls
Posted by [Mishin Dmitry](#) on Wed, 06 Dec 2006 13:06:08 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tuesday 05 December 2006 13:28, clg@fr.ibm.com wrote:

> From: Cedric Le Goater <clg@fr.ibm.com>

[skip]

> +static int switch_ns(int id, unsigned long flags)

> +{

> + int err = 0;

> + struct nsproxy *ns = NULL, *old_ns = NULL, *new_ns = NULL;

> +

> + if (flags & ~NS_ALL)

> + return -EINVAL;

> +

> + /* Let 0 be a default value ? */

> + if (!flags)

> + flags = NS_ALL;

> +

> + if (id < 0) {

> + struct task_struct *p;

> +

> + err = -ESRCH;

> + read_lock(&tasklist_lock);

> + p = find_task_by_pid(-id);

> + if (p) {

> + task_lock(p);

> + get_nsproxy(p->nsproxy);

> + ns = p->nsproxy;

> + task_unlock(p);

> + }

> + read_unlock(&tasklist_lock);

> + } else {

> + err = -ENOENT;

> + spin_lock_irq(&ns_hash_lock);

> + ns = ns_hash_find(id);

> + spin_unlock_irq(&ns_hash_lock);

> + }

> +

> + if (!ns)

> + goto out;

> +

> + new_ns = ns;

> +

> + /*

> + * clone current nsproxy and populate it with the namespaces

> + * chosen by flags.

> + */

```

> + if (flags != NS_ALL) {
> +     new_ns = dup_namespaces(current->nsproxy);
> +     if (!new_ns) {
> +         err = -ENOMEM;
> +         goto out_ns;
> +     }
> +
> +     if (flags & NS_MNT) {
> +         put_mnt_ns(new_ns->mnt_ns);
> +         get_mnt_ns(ns->mnt_ns);
> +         new_ns->mnt_ns = ns->mnt_ns;
> +     }
> +
> +     if (flags & NS_UTS) {
> +         put_uts_ns(new_ns->uts_ns);
> +         get_uts_ns(ns->uts_ns);
> +         new_ns->uts_ns = ns->uts_ns;
> +     }
> +
> +     if (flags & NS_IPC) {
> +         put_ipc_ns(new_ns->ipc_ns);
> +         new_ns->ipc_ns = get_ipc_ns(ns->ipc_ns);
> +     }
<<<< This code looks useless for me, as at this time new_ns->any_ptr ==
ns->any_ptr.

> + out_ns:
> +     put_nsproxy(ns);
> + }
> +
> + task_lock(current);
> + if (new_ns) {
> +     old_ns = current->nsproxy;
> +     current->nsproxy = new_ns;
> + }
> + task_unlock(current);
> +
> + if (old_ns)
> +     put_nsproxy(old_ns);
> +
> + err = 0;
> +out:
> + return err;
> +}
> +
> +
> +/*
> + * bind_ns - bind the nsproxy of a task to an id or bind a task to a

```

```

> + *      identified nsproxy
> + *
> + * @id: nsproxy identifier if positive or pid if negative
> + * @flags: identifies the namespaces to bind to
> + *
> + * bind_ns serves 2 purposes.
> + *
> + * The first is to bind the nsproxy of the current task to the
> + * identifier @id. If the identifier is already used, -EBUSY is
> + * returned. If the nsproxy is already bound, -EACCES is returned.
> + * flags is not used in that case.
> + *
> + * The second use is to bind the current task to a subset of
> + * namespaces of an identified nsproxy. If positive, @id is considered
> + * being an nsproxy identifier previously used to bind the nsproxy to
> + * @id. If negative, @id is the pid of a task which is another way to
> + * identify a nsproxy. Switching nsproxy is restricted to tasks within
> + * nsproxy 0, the default nsproxy. If unknown, -ENOENT is returned.
> + * @flags is used to bind the task to the selected namespaces.
> + *
> + * Both uses may return -EINVAL for invalid arguments and -EPERM for
> + * insufficient privileges.
> + *
> + * Returns 0 on success.
> + */
> +asmlinkage long sys_bind_ns(int id, unsigned long flags)
> +{
> + struct nsproxy *ns = current->nsproxy;
> + int ret = 0;
> +
> + /*
> + * ns is being changed by switch_ns(), protect it
> + */
> + get_nsproxy(ns);
> +
> + /*
> + * protect ns->id
> + */
> + spin_lock(&ns->nslock);
> + switch (ns->id) {
> + case -1:
> + /*
> + * only an unbound nsproxy can be bound to an id.
> + */
> + ret = bind_ns(id, ns);
> + break;
> +
> + case 0:

```

```

> + if (!capable(CAP_SYS_ADMIN)) {
> +     ret = -EPERM;
> +     goto unlock;
> + }
> +
> + /*
> +  * only nsproxy 0 can switch nsproxy. if target id is
> +  * 0, this is a nop.
> +  */
> + if (id)
> +     ret = switch_ns(id, flags);
> + break;
> +
> + default:
> +     /*
> +      * current nsproxy is already bound. forbid any
> +      * switch.
> +      */
> +     ret = -EACCES;
> + }
> + unlock:
> + spin_unlock(&ns->nslock);
> + put_nsproxy(ns);
> + return ret;
> +}
> +
> +/*
> + * sys_unshare_ns - unshare one or more of namespaces which were
> + *                   originally shared using clone.
> + *
> + * @unshare_ns_flags: identifies the namespaces to unshare
> + *
> + * this is for the moment a pale copy of unshare but we expect to
> + * diverge when the number of namespaces increases. the number of
> + * available clone flags is also very low. This is one way to get
> + * some air.
> + *
> + */
> +asmlinkage long sys_unshare_ns(unsigned long unshare_ns_flags)
> +{
> +     int err = 0;
> +     struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
> +     struct fs_struct *fs, *new_fs = NULL;
> +     struct mnt_namespace *mnt, *new_mnt = NULL;
> +     struct uts_namespace *uts, *new_uts = NULL;
> +     struct ipc_namespace *ipc, *new_ipc = NULL;
> +     unsigned long unshare_flags = 0;
> +

```

```

> + /* Return -EINVAL for all unsupported flags */
> + err = -EINVAL;
> + if (unshare_ns_flags & ~NS_ALL)
> +     goto bad_unshare_ns_out;
> +
> + /*
> +  * compatibility with unshare()/clone() : convert ns flags to
> +  * clone flags
> +  */
> + if (unshare_ns_flags & NS_MNT)
> +     unshare_flags |= CLONE_NEWNS|CLONE_FS;
> + if (unshare_ns_flags & NS_UTS)
> +     unshare_flags |= CLONE_NEWUTS;
> + if (unshare_ns_flags & NS_IPC)
> +     unshare_flags |= CLONE_NEWIPC;
> +
> + if ((err = unshare_fs(unshare_flags, &new_fs)))
> +     goto bad_unshare_ns_out;
> + if ((err = unshare_mnt_ns(unshare_flags, &new_mnt, new_fs)))
> +     goto bad_unshare_ns_cleanup_fs;
> + if ((err = unshare_utsname(unshare_flags, &new_uts)))
> +     goto bad_unshare_ns_cleanup_mnt;
> + if ((err = unshare_ipcs(unshare_flags, &new_ipc)))
> +     goto bad_unshare_ns_cleanup_uts;
> +
> + if (new_mnt || new_uts || new_ipc) {
> +     old_nsproxy = current->nsproxy;
> +     new_nsproxy = dup_namespaces(old_nsproxy);
> +     if (!new_nsproxy) {
> +         err = -ENOMEM;
> +         goto bad_unshare_ns_cleanup_ipc;
> +     }
> + }
> +
> + if (new_fs || new_mnt || new_uts || new_ipc) {
> +
> +     task_lock(current);
> +
> +     if (new_nsproxy) {
> +         current->nsproxy = new_nsproxy;
> +         new_nsproxy = old_nsproxy;
> +     }
> +
> +     if (new_fs) {
> +         fs = current->fs;
> +         current->fs = new_fs;
> +         new_fs = fs;
> +     }

```

```

> +
> + if (new_mnt) {
> +     mnt = current->nsproxy->mnt_ns;
> +     current->nsproxy->mnt_ns = new_mnt;
> +     new_mnt = mnt;
> + }
> +
> + if (new_uts) {
> +     uts = current->nsproxy->uts_ns;
> +     current->nsproxy->uts_ns = new_uts;
> +     new_uts = uts;
> + }
> +
> + if (new_ipc) {
> +     ipc = current->nsproxy->ipc_ns;
> +     current->nsproxy->ipc_ns = new_ipc;
> +     new_ipc = ipc;
> + }
> +
> + task_unlock(current);
> + }
> +
> + if (new_nsproxy)
> +     put_nsproxy(new_nsproxy);
> +
> +bad_unshare_ns_cleanup_ipc:
> + if (new_ipc)
> +     put_ipc_ns(new_ipc);
> +
> +bad_unshare_ns_cleanup_uts:
> + if (new_uts)
> +     put_uts_ns(new_uts);
> +
> +bad_unshare_ns_cleanup_mnt:
> + if (new_mnt)
> +     put_mnt_ns(new_mnt);
> +
> +bad_unshare_ns_cleanup_fs:
> + if (new_fs)
> +     put_fs_struct(new_fs);
> +
> +bad_unshare_ns_out:
> + return err;
> +}
> +
> static int __init nshash_init(void)
> {
>     int i;

```

```

> Index: 2.6.19-rc6-mm2/kernel/sys_ni.c
> =====
> --- 2.6.19-rc6-mm2.orig/kernel/sys_ni.c
> +++ 2.6.19-rc6-mm2/kernel/sys_ni.c
> @@ -146,3 +146,7 @@ cond_syscall(compat_sys_migrate_pages);
> cond_syscall(sys_bdflush);
> cond_syscall(sys_ioprio_set);
> cond_syscall(sys_ioprio_get);
> +
> +/* user resources syscalls */
> +cond_syscall(sys_unshare_ns);
> +cond_syscall(sys_bind_ns);
> Index: 2.6.19-rc6-mm2/arch/i386/kernel/syscall_table.S
> =====
> --- 2.6.19-rc6-mm2.orig/arch/i386/kernel/syscall_table.S
> +++ 2.6.19-rc6-mm2/arch/i386/kernel/syscall_table.S
> @@ -323,3 +323,5 @@ ENTRY(sys_call_table)
> .long sys_kevent_ctl
> .long sys_kevent_wait
> .long sys_kevent_ring_init
> + .long sys_unshare_ns
> + .long sys_bind_ns /* 325 */
> Index: 2.6.19-rc6-mm2/arch/ia64/kernel/entry.S
> =====
> --- 2.6.19-rc6-mm2.orig/arch/ia64/kernel/entry.S
> +++ 2.6.19-rc6-mm2/arch/ia64/kernel/entry.S
> @@ -1610,5 +1610,7 @@ sys_call_table:
> data8 sys_sync_file_range // 1300
> data8 sys_tee
> data8 sys_vmsplice
> + data8 sys_unshare_ns
> + data8 sys_bind_ns
>
> .org sys_call_table + 8*NR_syscalls // guard against failures to increase
> NR_syscalls Index: 2.6.19-rc6-mm2/arch/s390/kernel/compat_wrapper.S
> =====
> --- 2.6.19-rc6-mm2.orig/arch/s390/kernel/compat_wrapper.S
> +++ 2.6.19-rc6-mm2/arch/s390/kernel/compat_wrapper.S
> @@ -1665,3 +1665,14 @@ sys_getcpu_wrapper:
> llgr %r3,%r3 # unsigned *
> llgr %r4,%r4 # struct getcpu_cache *
> jg sys_getcpu
> +
> + .globl sys_unshare_ns_wrapper
> +sys_unshare_ns_wrapper:
> + llgr %r2,%r2 # unsigned long
> + jg sys_unshare_ns
> +

```

```

> + .globl sys_bind_ns_wrapper
> +sys_bind_ns_wrapper:
> + lgfr    %r2,%r2          # int
> + llgr    %r3,%r3          # unsigned long
> + jg      sys_bind_ns
> Index: 2.6.19-rc6-mm2/arch/s390/kernel/syscalls.S
> =====
> --- 2.6.19-rc6-mm2.orig/arch/s390/kernel/syscalls.S
> +++ 2.6.19-rc6-mm2/arch/s390/kernel/syscalls.S
> @@ -321,3 +321,5 @@ SYSCALL(sys_vmsplice,sys_vmsplice,compat
> NI_SYSCALL    /* 310 sys_move_pages */
> SYSCALL(sys_getcpu,sys_getcpu,sys_getcpu_wrapper)
> SYSCALL(sys_epoll_pwait,sys_epoll_pwait,sys_ni_syscall)
> +SYSCALL(sys_unshare_ns,sys_unshare_ns,sys_unshare_ns_wrappe r)
> +SYSCALL(sys_bind_ns,sys_bind_ns,sys_bind_ns_wrapper)
> Index: 2.6.19-rc6-mm2/arch/x86_64/ia32/ia32entry.S
> =====
> --- 2.6.19-rc6-mm2.orig/arch/x86_64/ia32/ia32entry.S
> +++ 2.6.19-rc6-mm2/arch/x86_64/ia32/ia32entry.S
> @@ -722,4 +722,6 @@ ia32_sys_call_table:
> .quad sys_kevent_ctl /* 320 */
> .quad sys_kevent_wait
> .quad sys_kevent_ring_init
> + .quad sys_unshare_ns
> + .quad sys_bind_ns
> ia32_syscall_end:
> Index: 2.6.19-rc6-mm2/include/asm-i386/unistd.h
> =====
> --- 2.6.19-rc6-mm2.orig/include/asm-i386/unistd.h
> +++ 2.6.19-rc6-mm2/include/asm-i386/unistd.h
> @@ -329,10 +329,12 @@
> #define __NR_kevent_ctl 321
> #define __NR_kevent_wait 322
> #define __NR_kevent_ring_init 323
> +#define __NR_unshare_ns 324
> +#define __NR_bind_ns 325
>
> #ifdef __KERNEL__
>
> -#define NR_syscalls 324
> +#define NR_syscalls 326
>
> #define __ARCH_WANT_IPC_PARSE_VERSION
> #define __ARCH_WANT_OLD_READDIR
> Index: 2.6.19-rc6-mm2/include/asm-ia64/unistd.h
> =====
> --- 2.6.19-rc6-mm2.orig/include/asm-ia64/unistd.h
> +++ 2.6.19-rc6-mm2/include/asm-ia64/unistd.h

```

```

> @@ -291,11 +291,13 @@
> #define __NR_sync_file_range 1300
> #define __NR_tee 1301
> #define __NR_vmsplice 1302
> +#define __NR_unshare_ns 1303
> +#define __NR_bind_ns 1304
>
> #ifdef __KERNEL__
>
>
> -#define NR_syscalls 279 /* length of syscall table */
> +#define NR_syscalls 281 /* length of syscall table */
>
> #define __ARCH_WANT_SYS_RT_SIGACTION
>
> Index: 2.6.19-rc6-mm2/include/asm-powerpc/systbl.h
> =====
> --- 2.6.19-rc6-mm2.orig/include/asm-powerpc/systbl.h
> +++ 2.6.19-rc6-mm2/include/asm-powerpc/systbl.h
> @@ -305,3 +305,5 @@ SYSCALL_SPU(faccessat)
> COMPAT_SYS_SPU(get_robust_list)
> COMPAT_SYS_SPU(set_robust_list)
> COMPAT_SYS(move_pages)
> +SYSCALL_SPU(unshare_ns)
> +SYSCALL_SPU(bind_ns)
> Index: 2.6.19-rc6-mm2/include/asm-powerpc/unistd.h
> =====
> --- 2.6.19-rc6-mm2.orig/include/asm-powerpc/unistd.h
> +++ 2.6.19-rc6-mm2/include/asm-powerpc/unistd.h
> @@ -324,10 +324,12 @@
> #define __NR_get_robust_list 299
> #define __NR_set_robust_list 300
> #define __NR_move_pages 301
> +#define __NR_unshare_ns 302
> +#define __NR_bind_ns 303
>
> #ifdef __KERNEL__
>
> -#define __NR_syscalls 302
> +#define __NR_syscalls 304
>
> #define __NR__exit __NR_exit
> #define NR_syscalls __NR_syscalls
> Index: 2.6.19-rc6-mm2/include/asm-s390/unistd.h
> =====
> --- 2.6.19-rc6-mm2.orig/include/asm-s390/unistd.h
> +++ 2.6.19-rc6-mm2/include/asm-s390/unistd.h
> @@ -250,8 +250,10 @@

```

```

> /* Number 310 is reserved for new sys_move_pages */
> #define __NR_getcpu 311
> #define __NR_epoll_pwait 312
> +#define __NR_unshare_ns 313
> +#define __NR_bind_ns 314
>
> -#define NR_syscalls 313
> +#define NR_syscalls 315
>
> /*
>  * There are some system calls that are not present on 64 bit, some
>  Index: 2.6.19-rc6-mm2/include/asm-x86_64/unistd.h
>  =====
>  --- 2.6.19-rc6-mm2.orig/include/asm-x86_64/unistd.h
>  +++ 2.6.19-rc6-mm2/include/asm-x86_64/unistd.h
>  @@ -627,8 +627,12 @@ __SYSCALL(__NR_kevent_ctl, sys_kevent_ct
>  __SYSCALL(__NR_kevent_wait, sys_kevent_wait)
>  #define __NR_kevent_ring_init 283
>  __SYSCALL(__NR_kevent_ring_init, sys_kevent_ring_init)
>  +#define __NR_unshare_ns 284
>  +__SYSCALL(__NR_unshare_ns, sys_unshare_ns)
>  +#define __NR_bind_ns 285
>  +__SYSCALL(__NR_bind_ns, sys_bind_ns)
>
> -#define __NR_syscall_max __NR_kevent_ring_init
> +#define __NR_syscall_max __NR_bind_ns
>
> #ifndef __NO_STUBS
> #define __ARCH_WANT_OLD_READDIR

```

--

Thanks,
Dmitry.

Subject: Re: [patch -mm 10/17] nsproxy: add unshare_ns and bind_ns syscalls
 Posted by [Daniel Lezcano](#) on Wed, 06 Dec 2006 21:01:32 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dmitry Mishin wrote:

```

> On Tuesday 05 December 2006 13:28, clg@fr.ibm.com wrote:
>> From: Cedric Le Goater <clg@fr.ibm.com>
> [skip]
>> +static int switch_ns(int id, unsigned long flags)
>> +{
>> + int err = 0;
>> + struct nsproxy *ns = NULL, *old_ns = NULL, *new_ns = NULL;
>> +

```

```

>> + if (flags & ~NS_ALL)
>> + return -EINVAL;
>> +
>> + /* Let 0 be a default value ? */
>> + if (!flags)
>> + flags = NS_ALL;
>> +
>> + if (id < 0) {
>> + struct task_struct *p;
>> +
>> + err = -ESRCH;
>> + read_lock(&tasklist_lock);
>> + p = find_task_by_pid(-id);
>> + if (p) {
>> + task_lock(p);
>> + get_nsproxy(p->nsproxy);
>> + ns = p->nsproxy;
>> + task_unlock(p);
>> + }
>> + read_unlock(&tasklist_lock);
>> + } else {
>> + err = -ENOENT;
>> + spin_lock_irq(&ns_hash_lock);
>> + ns = ns_hash_find(id);
>> + spin_unlock_irq(&ns_hash_lock);
>> + }
>> +
>> + if (!ns)
>> + goto out;
>> +
>> + new_ns = ns;
>> +
>> + /*
>> + * clone current nsproxy and populate it with the namespaces
>> + * chosen by flags.
>> + */
>> + if (flags != NS_ALL) {
>> + new_ns = dup_namespaces(current->nsproxy);
>> + if (!new_ns) {
>> + err = -ENOMEM;
>> + goto out_ns;
>> + }
>> +
>> + if (flags & NS_MNT) {
>> + put_mnt_ns(new_ns->mnt_ns);
>> + get_mnt_ns(ns->mnt_ns);
>> + new_ns->mnt_ns = ns->mnt_ns;
>> + }

```

```
>> +
>> + if (flags & NS_UTS) {
>> +   put_uts_ns(new_ns->uts_ns);
>> +   get_uts_ns(ns->uts_ns);
>> +   new_ns->uts_ns = ns->uts_ns;
>> + }
>> +
>> + if (flags & NS_IPC) {
>> +   put_ipc_ns(new_ns->ipc_ns);
>> +   new_ns->ipc_ns = get_ipc_ns(ns->ipc_ns);
>> + }
> <<<< This code looks useless for me, as at this time new_ns->any_ptr ==
> ns->any_ptr.
```

Yep. Because of the kmemdup in clone_namespace called by dup_namespace.
