Subject: Re: - merge-sys_clone-sys_unshare-nsproxy-and-namespace.patch removed from -mm tree
Posted by Cedric Le Goater on Mon, 18 Jun 2007 16:54:33 GMT
View Forum Message <> Reply to Message

[...]

>> It fixes the leak for me. I've run the ltp tests we
>> have on namespace unsharing and i could see the no
>> leaks in /proc/slabinfo.
>
>> Badari,
>>
>> That extra get_nsproxy() seemed a superfluous remain
>> from the 2.6.20.
>> Do you see any issues with it ?
>>
>> If we're all happy with these fixes, i'll send them on
>> lkml@ for review.
>
> I'm not terribly happy with the current nsproxy
> framework, although it improved somewhat ...
>
> I'm still missing some mechanism to 'mix' two
> proxies according to a flagmask (which is required
> to enter a guest 'partially') ...

We have that bind_ns() syscall that does that. I sent it last year
but it didn't have much success. We still use and there may be room
for improvement to make it altruistically useful.

Here's the patch on a 2.6.21-mm2. It it's of any interest, I can
refresh it on the latest -mm.

Thanks,

C.

The following patch defines the new bind_ns syscall specific to
nsproxy and namespaces, which allows a process to bind :

1 - its nsproxy to some identifier
2 - to another nsproxy using an identifier or -pid

Here's a sample user space program to use it.

```
#include <stdio.h>
#include <stdlib.h>
```

```
#include <sched.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>
#include <libgen.h>
#include <sys/syscall.h>

#ifndef HAVE_UNSHARE

#if __i386__
#    define __NR_unshare 310
#elif __x86_64__
#    define __NR_unshare 272
#elif __ia64__
#    define __NR_unshare 1296
#elif __s390x__
#    define __NR_unshare 303
#elif __powerpc__
#    define __NR_unshare 282
#else
#    error "Architecture not supported"
#endif

#endif /* HAVE_UNSHARE */

#if __i386__
#    define __NR_bind_ns  326
#elif __ia64__
#    define __NR_bind_ns 1305
#elif __powerpc__
#    define __NR_bind_ns 304
#elif __s390x__
#    define __NR_bind_ns 314
#elif __x86_64__
#    define __NR_bind_ns 284
#else
#    error "Architecture not supported"
#endif

#ifndef CLONE_NEWUTS
#define CLONE_NEWUTS  0x04000000
#endif

#ifndef CLONE_NEWIPC
#define CLONE_NEWIPC  0x08000000
#endif

#ifndef CLONE_NEWUSER
```

```
#define CLONE_NEWUSER  0x10000000
#endif

#ifndef CLONE_NEWNET2
#define CLONE_NEWNET2  0x20000000
#endif

#ifndef CLONE_NEWNET3
#define CLONE_NEWNET3  0x40000000
#endif

#ifndef CLONE_NEWPID
#define CLONE_NEWPID  0x80000000
#endif


static inline _syscall1 (int,  unshare, unsigned long, flags)
static inline _syscall2 (int,  bind_ns, int, id, unsigned long, flags)

static const char* procname;

static void usage(const char *name)
{
 printf("usage: %s [-h] [-I id] [-muiUnNp] [command [arg ...]]\n", name);
 printf("\n");
 printf("  -h  this message\n");
 printf("\n");
 printf("  -I <id> bind process to nsproxy <id>\n");
 printf("  -m  mount namespace\n");
 printf("  -u  utsname namespace\n");
 printf("  -i  ipc namespace\n");
 printf("  -U  user namespace\n");
 printf("  -n  net namespace level 2\n");
 printf("  -N  net namespace level 3\n");
 printf("  -p  pid namespace\n");
 printf("\n");
 printf("(C) Copyright IBM Corp. 2006\n");
 printf("\n");
 exit(1);
}

int main(int argc, char *argv[])
{
 int c;
 unsigned long flags = 0;
 int id = -1;

 procname = basename(argv[0]);
```

```
 while ((c = getopt(argc, argv, "+muiUnNphI:")) != EOF) {
  switch (c) {
  case 'I': if (optarg)
    id = atoi(optarg); break;

  case 'm': flags |= CLONE_NEWNS;  break;
  case 'u': flags |= CLONE_NEWUTS;  break;
  case 'i': flags |= CLONE_NEWIPC;  break;
  case 'U': flags |= CLONE_NEWUSER; break;
  case 'n': flags |= CLONE_NEWNET2;  break;
  case 'N': flags |= CLONE_NEWNET3;  break;
  case 'p': flags |= CLONE_NEWPID;  break;
  case 'h':
  default:
   usage(procname);
  }
 };

 argv = &argv[optind];
 argc = argc - optind;

 if (!strcmp(procname, "unsharens")) {
  if (unshare(flags) == -1) {
   perror("unshare");
   return 1;
  }
 }

 if (bind_ns(id, flags) == -1) {
  perror("bind_ns");
  return 1;
 }

 if (argc) {
  execve(argv[0], argv, __environ);
  perror("execve");
  return 1;
 }

 return 0;
}
```

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

```
 arch/i386/kernel/syscall_table.S  |    1
 arch/ia64/kernel/entry.S          |    1
```

```
 arch/s390/kernel/compat_wrapper.S |    6
 arch/s390/kernel/syscalls.S       |    1
 arch/x86_64/ia32/ia32entry.S      |    1
 include/asm-i386/unistd.h         |    3
 include/asm-ia64/unistd.h         |    3
 include/asm-powerpc/systbl.h      |    1
 include/asm-powerpc/unistd.h      |    3
 include/asm-s390/unistd.h         |    3
 include/asm-x86_64/unistd.h       |    2
 include/linux/nsproxy.h           |   10 -
 include/linux/sched.h             |    2
 include/linux/syscalls.h          |    2
 kernel/nsproxy.c                  |  264 ++++++++++++++++++++++++++++++++++++++-
 kernel/sys_ni.c                   |    2
 16 files changed, 290 insertions(+), 15 deletions(-)

Index: 2.6.21-mm2/include/linux/syscalls.h
===================================================================
--- 2.6.21-mm2.orig/include/linux/syscalls.h
+++ 2.6.21-mm2/include/linux/syscalls.h
@@ -609,6 +609,8 @@ asmlinkage long sys_timerfd(int ufd, int
        const struct itimerspec __user *utmr);
 asmlinkage long sys_eventfd(unsigned int count);

+asmlinkage long sys_bind_ns(int id, unsigned long unshare_flags);
+
 int kernel_execve(const char *filename, char *const argv[], char *const envp[]);

 asmlinkage long sys_revokeat(int dfd, const char __user *filename);
Index: 2.6.21-mm2/kernel/nsproxy.c
===================================================================
--- 2.6.21-mm2.orig/kernel/nsproxy.c
+++ 2.6.21-mm2/kernel/nsproxy.c
@@ -22,7 +22,11 @@

 static struct kmem_cache *nsproxy_cachep;

-#define NS_HASH_BITS   3 /* this might need some configuration */
+/*
+ * nsproxies are stored in a hash but a rbtree might be more
+ * appropriate.
+ */
+#define NS_HASH_BITS   3
 #define NS_HASH_SIZE  (1 << NS_HASH_BITS)
 #define NS_HASH_MASK  (NS_HASH_SIZE - 1)
 #define ns_hashfn(id)  (((id >> NS_HASH_BITS) + id) & NS_HASH_MASK)
@@ -63,6 +67,30 @@ static inline struct nsproxy *clone_nspr
 }
```

```
 /*
+ * copies the nsproxy, setting refcount to 1, and grabbing a
+ * reference to all contained namespaces.  Called from
+ * sys_unshare()
+ */
+static struct nsproxy *dup_namespaces(struct nsproxy *orig)
+{
+ struct nsproxy *ns = clone_nsproxy(orig);
+
+ if (ns) {
+  if (ns->mnt_ns)
+   get_mnt_ns(ns->mnt_ns);
+  if (ns->uts_ns)
+   get_uts_ns(ns->uts_ns);
+  if (ns->ipc_ns)
+   get_ipc_ns(ns->ipc_ns);
+  if (ns->pid_ns)
+   get_pid_ns(ns->pid_ns);
+  if (ns->user_ns)
+   get_user_ns(ns->user_ns);
+ }
+
+ return ns;
+}
+/*
  * Create new nsproxy and all of its the associated namespaces.
  * Return the newly created nsproxy.  Do not attach this to the task,
  * leave it to the caller to do proper locking and attach it to task.
@@ -123,7 +151,7 @@ int copy_namespaces(int flags, struct ta

  get_nsproxy(old_ns);

- if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC | CLONE_NEWUSER)))
+ if (!(flags & NS_ALL))
   return 0;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -143,7 +171,7 @@ out:
  return err;
 }

-void free_nsproxy(struct nsproxy *ns)
+static void free_nsproxy(struct nsproxy *ns)
 {
  if (ns->mnt_ns)
   put_mnt_ns(ns->mnt_ns);
@@ -157,6 +185,29 @@ void free_nsproxy(struct nsproxy *ns)
```

```
 }

 /*
+ * put_nsproxy() is similar to free_uid() in kernel/user.c
+ *
+ * the lock can be taken from a tasklet context (task getting freed by
+ * RCU) which requires to be irq safe.
+ */
+void put_nsproxy(struct nsproxy *ns)
+{
+ unsigned long flags;
+
+ local_irq_save(flags);
+ if (atomic_dec_and_lock(&ns->count, &ns_hash_lock)) {
+  BUG_ON(!ns->id);
+  if (ns->id != -1)
+   hlist_del(&ns->ns_hash_node);
+  spin_unlock_irqrestore(&ns_hash_lock, flags);
+  free_nsproxy(ns);
+ } else {
+  local_irq_restore(flags);
+ }
+}
+EXPORT_SYMBOL_GPL(put_nsproxy);
+
+/*
  * Called from unshare. Unshare all the namespaces part of nsproxy.
  * On sucess, returns the new nsproxy and a reference to old nsproxy
  * to make sure it stays around.
@@ -211,6 +262,212 @@ static inline struct nsproxy *ns_hash_fi
  return NULL;
 }

+struct nsproxy *find_nsproxy_by_id(int id)
+{
+  struct nsproxy *ns;
+
+  if (id < 0)
+   return NULL;
+
+  spin_lock_irq(&ns_hash_lock);
+  ns = ns_hash_find(id);
+ spin_unlock_irq(&ns_hash_lock);
+
+  return ns;
+}
+
+EXPORT_SYMBOL_GPL(find_nsproxy_by_id);
```

```
+
+static int bind_ns(int id, struct nsproxy *ns)
+{
+  struct nsproxy *prev;
+ int ret = 0;
+
+  if (id < 0)
+   return -EINVAL;
+
+  spin_lock_irq(&ns_hash_lock);
+  prev = ns_hash_find(id);
+  if (!prev) {
+  ns->id = id;
+  hlist_add_head(&ns->ns_hash_node, ns_hash_head(ns->id));
+ }
+ spin_unlock_irq(&ns_hash_lock);
+
+ if (prev) {
+  ret = -EBUSY;
+  put_nsproxy(prev);
+ }
+  return ret;
+}
+
+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 & CLONE_NEWNS) {
+   put_mnt_ns(new_ns->mnt_ns);
+   get_mnt_ns(ns->mnt_ns);
+   new_ns->mnt_ns = ns->mnt_ns;
+  }
+
+  if (flags & CLONE_NEWUTS) {
+   put_uts_ns(new_ns->uts_ns);
+   get_uts_ns(ns->uts_ns);
+   new_ns->uts_ns = ns->uts_ns;
+  }
+
+  if (flags & CLONE_NEWIPC) {
+   put_ipc_ns(new_ns->ipc_ns);
+   new_ns->ipc_ns = get_ipc_ns(ns->ipc_ns);
+  }
+
+  if (flags & CLONE_NEWUSER) {
+   put_user_ns(new_ns->user_ns);
+   get_user_ns(ns->user_ns);
+   new_ns->user_ns = ns->user_ns;
+  }
+
+ 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;
+}
+
 static int __init nshash_init(void)
```

```
 {
  int i;
@@ -235,3 +492,4 @@ static int __init nsproxy_cache_init(voi
 }

 module_init(nsproxy_cache_init);
+
Index: 2.6.21-mm2/kernel/sys_ni.c
===================================================================
--- 2.6.21-mm2.orig/kernel/sys_ni.c
+++ 2.6.21-mm2/kernel/sys_ni.c
@@ -146,3 +146,5 @@ cond_syscall(sys_ioprio_get);
 cond_syscall(sys_signalfd);
 cond_syscall(sys_timerfd);
 cond_syscall(sys_eventfd);
+
+cond_syscall(sys_bind_ns);
Index: 2.6.21-mm2/arch/ia64/kernel/entry.S
===================================================================
--- 2.6.21-mm2.orig/arch/ia64/kernel/entry.S
+++ 2.6.21-mm2/arch/ia64/kernel/entry.S
@@ -1583,5 +1583,6 @@ sys_call_table:
  data8 sys_vmsplice
  data8 sys_ni_syscall   // reserved for move_pages
  data8 sys_getcpu
+ data8 sys_bind_ns

  .org sys_call_table + 8*NR_syscalls // guard against failures to increase NR_syscalls
Index: 2.6.21-mm2/arch/s390/kernel/compat_wrapper.S
===================================================================
--- 2.6.21-mm2.orig/arch/s390/kernel/compat_wrapper.S
+++ 2.6.21-mm2/arch/s390/kernel/compat_wrapper.S
@@ -1682,3 +1682,9 @@ compat_sys_utimes_wrapper:
  llgtr %r2,%r2   # char *
  llgtr %r3,%r3   # struct compat_timeval *
  jg compat_sys_utimes
+
+ .globl sys_bind_ns_wrapper
+sys_bind_ns_wrapper:
+ lgfr    %r2,%r2                 # int
+ llgfr   %r3,%r3                 # unsigned long
+ jg      sys_bind_ns
Index: 2.6.21-mm2/arch/s390/kernel/syscalls.S
===================================================================
--- 2.6.21-mm2.orig/arch/s390/kernel/syscalls.S
+++ 2.6.21-mm2/arch/s390/kernel/syscalls.S
@@ -322,3 +322,4 @@ NI_SYSCALL       /* 310 sys_move_pages *
 SYSCALL(sys_getcpu,sys_getcpu,sys_getcpu_wrapper)
```

```
 SYSCALL(sys_epoll_pwait,sys_epoll_pwait,compat_sys_epoll_pwait_wrapper)
 SYSCALL(sys_utimes,sys_utimes,compat_sys_utimes_wrapper)
+SYSCALL(sys_bind_ns,sys_bind_ns,sys_bind_ns_wrapper)
Index: 2.6.21-mm2/arch/x86_64/ia32/ia32entry.S
===================================================================
--- 2.6.21-mm2.orig/arch/x86_64/ia32/ia32entry.S
+++ 2.6.21-mm2/arch/x86_64/ia32/ia32entry.S
@@ -721,4 +721,5 @@ ia32_sys_call_table:
  .quad sys_eventfd
  .quad sys_revokeat
  .quad sys_frevoke  /* 325 */
+ .quad sys_bind_ns
 ia32_syscall_end:
Index: 2.6.21-mm2/include/asm-ia64/unistd.h
===================================================================
--- 2.6.21-mm2.orig/include/asm-ia64/unistd.h
+++ 2.6.21-mm2/include/asm-ia64/unistd.h
@@ -294,11 +294,12 @@
 #define __NR_vmsplice   1302
 /* 1303 reserved for move_pages */
 #define __NR_getcpu   1304
+#define __NR_bind_ns   1305

 #ifdef __KERNEL__


-#define NR_syscalls   281 /* length of syscall table */
+#define NR_syscalls   282 /* length of syscall table */

 #define __ARCH_WANT_SYS_RT_SIGACTION
 #define __ARCH_WANT_SYS_RT_SIGSUSPEND
Index: 2.6.21-mm2/include/asm-powerpc/systbl.h
===================================================================
--- 2.6.21-mm2.orig/include/asm-powerpc/systbl.h
+++ 2.6.21-mm2/include/asm-powerpc/systbl.h
@@ -307,3 +307,4 @@ COMPAT_SYS_SPU(set_robust_list)
 COMPAT_SYS_SPU(move_pages)
 SYSCALL_SPU(getcpu)
 COMPAT_SYS(epoll_pwait)
+SYSCALL_SPU(bind_ns)
Index: 2.6.21-mm2/include/asm-powerpc/unistd.h
===================================================================
--- 2.6.21-mm2.orig/include/asm-powerpc/unistd.h
+++ 2.6.21-mm2/include/asm-powerpc/unistd.h
@@ -326,10 +326,11 @@
 #define __NR_move_pages  301
 #define __NR_getcpu  302
 #define __NR_epoll_pwait 303
```

```
+#define __NR_bind_ns  304

 #ifdef __KERNEL__

-#define __NR_syscalls  304
+#define __NR_syscalls  305

 #define __NR__exit __NR_exit
 #define NR_syscalls __NR_syscalls
Index: 2.6.21-mm2/include/asm-s390/unistd.h
===================================================================
--- 2.6.21-mm2.orig/include/asm-s390/unistd.h
+++ 2.6.21-mm2/include/asm-s390/unistd.h
@@ -251,8 +251,9 @@
 #define __NR_getcpu  311
 #define __NR_epoll_pwait 312
 #define __NR_utimes  313
+#define __NR_bind_ns  314

-#define NR_syscalls 314
+#define NR_syscalls 315

 /*
  * There are some system calls that are not present on 64 bit, some
Index: 2.6.21-mm2/include/asm-x86_64/unistd.h
===================================================================
--- 2.6.21-mm2.orig/include/asm-x86_64/unistd.h
+++ 2.6.21-mm2/include/asm-x86_64/unistd.h
@@ -627,6 +627,8 @@ __SYSCALL(__NR_signalfd, sys_signalfd)
 __SYSCALL(__NR_timerfd, sys_timerfd)
 #define __NR_eventfd  283
 __SYSCALL(__NR_eventfd, sys_eventfd)
+#define __NR_bind_ns  284
+__SYSCALL(__NR_bind_ns, sys_bind_ns)

 #ifndef __NO_STUBS
 #define __ARCH_WANT_OLD_READDIR
Index: 2.6.21-mm2/include/linux/sched.h
===================================================================
--- 2.6.21-mm2.orig/include/linux/sched.h
+++ 2.6.21-mm2/include/linux/sched.h
@@ -29,6 +29,8 @@
 #define CLONE_NEWPID  0x10000000      /* New pid namespace */
 #define CLONE_NEWUSER  0x20000000      /* New user namespace */

+#define NS_ALL (CLONE_NEWNS|CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER)
+
 /*
```

```
  * Scheduling policies
  */
Index: 2.6.21-mm2/arch/i386/kernel/syscall_table.S
===================================================================
--- 2.6.21-mm2.orig/arch/i386/kernel/syscall_table.S
+++ 2.6.21-mm2/arch/i386/kernel/syscall_table.S
@@ -325,3 +325,4 @@ ENTRY(sys_call_table)
  .long sys_eventfd
  .long sys_revokeat
  .long sys_frevoke  /* 325 */
+ .long sys_bind_ns
Index: 2.6.21-mm2/include/asm-i386/unistd.h
===================================================================
--- 2.6.21-mm2.orig/include/asm-i386/unistd.h
+++ 2.6.21-mm2/include/asm-i386/unistd.h
@@ -331,10 +331,11 @@
 #define __NR_eventfd  323
 #define __NR_revokeat  324
 #define __NR_frevoke  325
+#define __NR_bind_ns  326

 #ifdef __KERNEL__

-#define NR_syscalls 326
+#define NR_syscalls 327

 #define __ARCH_WANT_IPC_PARSE_VERSION
 #define __ARCH_WANT_OLD_READDIR
Index: 2.6.21-mm2/include/linux/nsproxy.h
===================================================================
--- 2.6.21-mm2.orig/include/linux/nsproxy.h
+++ 2.6.21-mm2/include/linux/nsproxy.h
@@ -36,16 +36,10 @@ extern struct nsproxy init_nsproxy;

 int copy_namespaces(int flags, struct task_struct *tsk);
 void get_task_nsproxy(struct task_struct *tsk);
-void free_nsproxy(struct nsproxy *ns);
 int unshare_nsproxy_namespaces(unsigned long, struct nsproxy **,
  struct fs_struct *);
-
-static inline void put_nsproxy(struct nsproxy *ns)
-{
- if (atomic_dec_and_test(&ns->count)) {
-  free_nsproxy(ns);
- }
-}
+void put_nsproxy(struct nsproxy *ns);
+struct nsproxy *find_nsproxy_by_id(int id);
```

static inline void put_task_nsproxy(struct task_struct *p)
{

_____

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