
Subject: [PATCH] Merge sys_clone() and sys_unshare() nsproxy handling
Posted by [Badari Pulavarty](#) on Tue, 27 Feb 2007 00:40:16 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi,

sys_clone() and sys_unshare() handles copy/clone of nsproxy and its associated namespaces differently.

This patch merges all the handling into common functions (as much as possible).

Please review. If no objections, I would like to submit for inclusion into -mm.

Ran standard LTP tests, unshare uts, ipc tests fine.

Thanks,
Badari

Merge sys_clone/sys_unshare nsproxy and namespace copy handling.

- Create a new nsproxy and its associated namespaces and pass it back to caller to attach it to right process.

- Changed all copy_*_ns() routines to attach the namespace to passed in new nsproxy instead of task->nsproxy.

- Get rid of all individual unshare_*_ns() routines and make use of copy_*_ns() instead.

Signed-off-by: Badari Pulavarty <pbadari@us.ibm.com>

```
fs/namespace.c      | 11 +--
include/linux/ipc.h  |  8 +-
include/linux/mnt_namespace.h |  5 -
include/linux/nsproxy.h |  2
include/linux/pid_namespace.h |  3
include/linux/utsname.h | 16 -----
ipc/util.c          | 24 -----
kernel/fork.c        | 85 +-----
kernel/nsproxy.c     | 133 ++++++-----
kernel/pid.c         |  4 -
kernel/utsname.c     | 24 -----
11 files changed, 113 insertions(+), 202 deletions(-)
```

Index: linux-2.6.21-rc1/fs/namespace.c

```

=====
--- linux-2.6.21-rc1.orig/fs/namespace.c 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/fs/namespace.c 2007-02-26 10:27:42.000000000 -0800
@@ -1441,10 +1441,9 @@ dput_out:
    * Allocate a new namespace structure and populate it with contents
    * copied from the namespace of the passed in task structure.
    */
-struct mnt_namespace *dup_mnt_ns(struct task_struct *tsk,
+static struct mnt_namespace *dup_mnt_ns(struct mnt_namespace *mnt_ns,
    struct fs_struct *fs)
{
- struct mnt_namespace *mnt_ns = tsk->nsproxy->mnt_ns;
  struct mnt_namespace *new_ns;
  struct vfsmount *rootmnt = NULL, *pwmnt = NULL, *altrootmnt = NULL;
  struct vfsmount *p, *q;
@@ -1509,9 +1508,9 @@ struct mnt_namespace *dup_mnt_ns(struct
    return new_ns;
}

-int copy_mnt_ns(int flags, struct task_struct *tsk)
+int copy_mnt_ns(int flags, struct mnt_namespace *ns, struct nsproxy *new_nsp,
+ struct fs_struct *new_fs)
{
- struct mnt_namespace *ns = tsk->nsproxy->mnt_ns;
  struct mnt_namespace *new_ns;
  int err = 0;

@@ -1528,13 +1527,13 @@ int copy_mnt_ns(int flags, struct task_s
    goto out;
}

- new_ns = dup_mnt_ns(tsk, tsk->fs);
+ new_ns = dup_mnt_ns(ns, new_fs);
  if (!new_ns) {
    err = -ENOMEM;
    goto out;
  }

- tsk->nsproxy->mnt_ns = new_ns;
+ new_nsp->mnt_ns = new_ns;

  out:
    put_mnt_ns(ns);
Index: linux-2.6.21-rc1/include/linux/mnt_namespace.h
=====
--- linux-2.6.21-rc1.orig/include/linux/mnt_namespace.h 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/include/linux/mnt_namespace.h 2007-02-26 10:27:42.000000000 -0800
@@ -14,10 +14,9 @@ struct mnt_namespace {

```

```

    int event;
};

-extern int copy_mnt_ns(int, struct task_struct *);
-extern void __put_mnt_ns(struct mnt_namespace *ns);
-extern struct mnt_namespace *dup_mnt_ns(struct task_struct *,
+extern int copy_mnt_ns(int, struct mnt_namespace *, struct nsproxy *,
    struct fs_struct *);
+extern void __put_mnt_ns(struct mnt_namespace *ns);

static inline void put_mnt_ns(struct mnt_namespace *ns)
{
Index: linux-2.6.21-rc1/kernel/nsproxy.c
=====
--- linux-2.6.21-rc1.orig/kernel/nsproxy.c 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/kernel/nsproxy.c 2007-02-26 10:27:42.000000000 -0800
@@ -38,8 +38,6 @@ void get_task_namespaces(struct task_str

/*
 * creates a copy of "orig" with refcount 1.
- * This does not grab references to the contained namespaces,
- * so that needs to be done by dup_namespaces.
 */
static inline struct nsproxy *clone_namespaces(struct nsproxy *orig)
{
@@ -52,26 +50,50 @@ static inline struct nsproxy *clone_name
}

/*
- * copies the nsproxy, setting refcount to 1, and grabbing a
- * reference to all contained namespaces. Called from
- * sys_unshare()
+ * 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.
 */
-struct nsproxy *dup_namespaces(struct nsproxy *orig)
+static struct nsproxy *create_new_namespaces(int flags, struct task_struct *tsk,
+ struct fs_struct *new_fs)
{
- struct nsproxy *ns = clone_namespaces(orig);
+ struct nsproxy *new_nsp;
+ int err;

- 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);
- }
+ new_nsp = clone_namespaces(tsk->nsproxy);
+ if (!new_nsp)
+ return ERR_PTR(-ENOMEM);

- return ns;
+ err = copy_mnt_ns(flags, tsk->nsproxy->mnt_ns, new_nsp, new_fs);
+ if (err)
+ goto out_ns;
+
+ err = copy_utsname(flags, tsk->nsproxy->uts_ns, new_nsp);
+ if (err)
+ goto out_uts;
+
+ err = copy_ipcs(flags, tsk->nsproxy->ipc_ns, new_nsp);
+ if (err)
+ goto out_ipc;
+
+ err = copy_pid_ns(flags, tsk->nsproxy->pid_ns, new_nsp);
+ if (err)
+ goto out_pid;
+
+ return new_nsp;
+
+out_pid:
+ if (new_nsp->ipc_ns)
+ put_ipc_ns(new_nsp->ipc_ns);
+out_ipc:
+ if (new_nsp->uts_ns)
+ put_uts_ns(new_nsp->uts_ns);
+out_uts:
+ if (new_nsp->mnt_ns)
+ put_mnt_ns(new_nsp->mnt_ns);
+out_ns:
+ kfree(new_nsp);
+ return ERR_PTR(err);
}

/*
@@ -92,47 +114,16 @@ int copy_namespaces(int flags, struct ta
if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
return 0;

```

```

- new_ns = clone_namespaces(old_ns);
- if (!new_ns) {
-   err = -ENOMEM;
+ new_ns = create_new_namespaces(flags, tsk, tsk->fs);
+ if (IS_ERR(new_ns)) {
+   err = PTR_ERR(new_ns);
+   goto out;
+ }

    tsk->nsproxy = new_ns;
-
- err = copy_mnt_ns(flags, tsk);
- if (err)
-   goto out_ns;
-
- err = copy_utsname(flags, tsk);
- if (err)
-   goto out_uts;
-
- err = copy_ipcs(flags, tsk);
- if (err)
-   goto out_ipc;
-
- err = copy_pid_ns(flags, tsk);
- if (err)
-   goto out_pid;
-
out:
    put_nsproxy(old_ns);
    return err;
-
-out_pid:
- if (new_ns->ipc_ns)
-   put_ipc_ns(new_ns->ipc_ns);
-out_ipc:
- if (new_ns->uts_ns)
-   put_uts_ns(new_ns->uts_ns);
-out_uts:
- if (new_ns->mnt_ns)
-   put_mnt_ns(new_ns->mnt_ns);
-out_ns:
- tsk->nsproxy = old_ns;
- kfree(new_ns);
- goto out;
- }

void free_nsproxy(struct nsproxy *ns)
@@ -147,3 +138,41 @@ void free_nsproxy(struct nsproxy *ns)

```

```

    put_pid_ns(ns->pid_ns);
    kfree(ns);
}
+
+/*
+ * Called from unshare. Unshare all the namespaces part of nsproxy.
+ * On success, returns the new nsproxy and a reference to old nsproxy
+ * to make sure it stays around.
+ */
+int unshare_nsproxy_namespaces(unsigned long unshare_flags,
+ struct nsproxy **new_nsp, struct fs_struct *new_fs)
+{
+ struct nsproxy *old_ns = current->nsproxy;
+ int err = 0;
+
+ if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC)))
+ return 0;
+
+ #ifndef CONFIG_IPC_NS
+ if (unshare_flags & CLONE_NEWIPC)
+ return -EINVAL;
+ #endif
+
+ #ifndef CONFIG_UTS_NS
+ if (unshare_flags & CLONE_NEWUTS)
+ return -EINVAL;
+ #endif
+
+ if (!capable(CAP_SYS_ADMIN))
+ return -EPERM;
+
+ get_nsproxy(old_ns);
+
+ *new_nsp = create_new_namespaces(unshare_flags, current,
+ new_fs ? new_fs : current->fs);
+ if (IS_ERR(*new_nsp)) {
+ err = PTR_ERR(*new_nsp);
+ put_nsproxy(old_ns);
+ }
+ return err;
+}

```

Index: linux-2.6.21-rc1/include/linux/utsname.h

```

=====
--- linux-2.6.21-rc1.orig/include/linux/utsname.h 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/include/linux/utsname.h 2007-02-26 10:27:42.000000000 -0800
@@ -49,9 +49,7 @@ static inline void get_uts_ns(struct uts
}

```

```

#ifdef CONFIG_UTS_NS
-extern int unshare_utsname(unsigned long unshare_flags,
- struct uts_namespace **new_uts);
-extern int copy_utsname(int flags, struct task_struct *tsk);
+extern int copy_utsname(int flags, struct uts_namespace *ns, struct nsproxy *);
extern void free_uts_ns(struct kref *kref);

static inline void put_uts_ns(struct uts_namespace *ns)
@@ -59,16 +57,8 @@ static inline void put_uts_ns(struct uts
    kref_put(&ns->kref, free_uts_ns);
}
#else
-static inline int unshare_utsname(unsigned long unshare_flags,
- struct uts_namespace **new_uts)
-{
- if (unshare_flags & CLONE_NEWUTS)
- return -EINVAL;
-
- return 0;
-}
-
-static inline int copy_utsname(int flags, struct task_struct *tsk)
+static inline int copy_utsname(int flags, struct uts_namespace *ns,
+ struct nsproxy *nsp)
{
    return 0;
}
Index: linux-2.6.21-rc1/kernel/utsname.c
=====
--- linux-2.6.21-rc1.orig/kernel/utsname.c 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/kernel/utsname.c 2007-02-26 10:27:42.000000000 -0800
@@ -32,32 +32,14 @@ static struct uts_namespace *clone_uts_n
}

/*
- * unshare the current process' utsname namespace.
- * called only in sys_unshare()
- */
-int unshare_utsname(unsigned long unshare_flags, struct uts_namespace **new_uts)
-{
- if (unshare_flags & CLONE_NEWUTS) {
- if (!capable(CAP_SYS_ADMIN))
- return -EPERM;
-
- *new_uts = clone_uts_ns(current->nsproxy->uts_ns);
- if (!*new_uts)
- return -ENOMEM;
- }
-}

```

```

-
- return 0;
-}
-
-/*
 * Copy task tsk's utsname namespace, or clone it if flags
 * specifies CLONE_NEWUTS. In latter case, changes to the
 * utsname of this process won't be seen by parent, and vice
 * versa.
 */
-int copy_utsname(int flags, struct task_struct *tsk)
+int copy_utsname(int flags, struct uts_namespace *old_ns,
+ struct nsproxy *new_nsp)
{
- struct uts_namespace *old_ns = tsk->nsproxy->uts_ns;
  struct uts_namespace *new_ns;
  int err = 0;

@@ -79,7 +61,7 @@ int copy_utsname(int flags, struct task_
  err = -ENOMEM;
  goto out;
}
- tsk->nsproxy->uts_ns = new_ns;
+ new_nsp->uts_ns = new_ns;

out:
  put_uts_ns(old_ns);
Index: linux-2.6.21-rc1/include/linux/ipc.h
=====
--- linux-2.6.21-rc1.orig/include/linux/ipc.h 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/include/linux/ipc.h 2007-02-26 10:27:42.000000000 -0800
@@ -71,6 +71,7 @@ struct kern_ipc_perm
};

struct ipc_ids;
+struct nsproxy;
struct ipc_namespace {
  struct kref kref;
  struct ipc_ids *ids[3];
@@ -98,10 +99,11 @@ extern struct ipc_namespace init_ipc_ns;

#ifdef CONFIG_IPC_NS
extern void free_ipc_ns(struct kref *kref);
-extern int copy_ipcs(unsigned long flags, struct task_struct *tsk);
-extern int unshare_ipcs(unsigned long flags, struct ipc_namespace **ns);
+extern int copy_ipcs(unsigned long flags, struct ipc_namespace *ns,
+  struct nsproxy *new_nsp);
#else

```



```

-static inline int copy_ipcs(unsigned long flags, struct task_struct *tsk)
+static inline int copy_ipcs(unsigned long flags, struct ipc_namespace *ns,
+ struct nsproxy *new_nsp)
{
    return 0;
}

```

Index: linux-2.6.21-rc1/ipc/util.c

```

=====
--- linux-2.6.21-rc1.orig/ipc/util.c 2007-02-20 20:32:30.000000000 -0800
+++ linux-2.6.21-rc1/ipc/util.c 2007-02-26 10:27:42.000000000 -0800
@@ -85,27 +85,9 @@ err_mem:
     return ERR_PTR(err);
}

```

```

-int unshare_ipcs(unsigned long unshare_flags, struct ipc_namespace **new_ipc)
+int copy_ipcs(unsigned long flags, struct ipc_namespace *old_ns,
+ struct nsproxy *new_nsp)
{
- struct ipc_namespace *new;
-
- if (unshare_flags & CLONE_NEWIPC) {
-     if (!capable(CAP_SYS_ADMIN))
-         return -EPERM;
-
-     new = clone_ipc_ns(current->nsproxy->ipc_ns);
-     if (IS_ERR(new))
-         return PTR_ERR(new);
-
-     *new_ipc = new;
- }
-
- return 0;
-}

```

```

-int copy_ipcs(unsigned long flags, struct task_struct *tsk)
-{
- struct ipc_namespace *old_ns = tsk->nsproxy->ipc_ns;
- struct ipc_namespace *new_ns;
- int err = 0;

```

```

@@ -128,7 +110,7 @@ int copy_ipcs(unsigned long flags, struc
    goto out;
}

```

```

- tsk->nsproxy->ipc_ns = new_ns;
+ new_nsp->ipc_ns = new_ns;
out:
    put_ipc_ns(old_ns);

```

```
return err;
```

```
Index: linux-2.6.21-rc1/include/linux/pid_namespace.h
```

```
-----  
--- linux-2.6.21-rc1.orig/include/linux/pid_namespace.h 2007-02-20 20:32:30.000000000 -0800  
+++ linux-2.6.21-rc1/include/linux/pid_namespace.h 2007-02-26 10:27:42.000000000 -0800  
@@ -29,7 +29,8 @@ static inline void get_pid_ns(struct pid  
    kref_get(&ns->kref);  
}
```

```
-extern int copy_pid_ns(int flags, struct task_struct *tsk);  
+extern int copy_pid_ns(int flags, struct pid_namespace *ns,  
+ struct nsproxy *nsp);  
extern void free_pid_ns(struct kref *kref);
```

```
static inline void put_pid_ns(struct pid_namespace *ns)
```

```
Index: linux-2.6.21-rc1/kernel/pid.c
```

```
-----  
--- linux-2.6.21-rc1.orig/kernel/pid.c 2007-02-20 20:32:30.000000000 -0800  
+++ linux-2.6.21-rc1/kernel/pid.c 2007-02-26 10:27:42.000000000 -0800  
@@ -360,9 +360,9 @@ struct pid *find_ge_pid(int nr)  
}  
EXPORT_SYMBOL_GPL(find_get_pid);
```

```
-int copy_pid_ns(int flags, struct task_struct *tsk)  
+int copy_pid_ns(int flags, struct pid_namespace *old_ns,  
+ struct nsproxy *new_nsp)  
{  
- struct pid_namespace *old_ns = tsk->nsproxy->pid_ns;  
  int err = 0;
```

```
    if (!old_ns)
```

```
Index: linux-2.6.21-rc1/include/linux/nsproxy.h
```

```
-----  
--- linux-2.6.21-rc1.orig/include/linux/nsproxy.h 2007-02-20 20:32:30.000000000 -0800  
+++ linux-2.6.21-rc1/include/linux/nsproxy.h 2007-02-26 10:27:42.000000000 -0800  
@@ -35,6 +35,8 @@ struct nsproxy *dup_namespaces(struct ns  
    int copy_namespaces(int flags, struct task_struct *tsk);  
    void get_task_namespaces(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)
```

```
{
```

```
Index: linux-2.6.21-rc1/kernel/fork.c
```

```
-----  
--- linux-2.6.21-rc1.orig/kernel/fork.c 2007-02-20 20:32:30.000000000 -0800  
+++ linux-2.6.21-rc1/kernel/fork.c 2007-02-26 10:27:42.000000000 -0800
```

```

@@ -1515,26 +1515,6 @@ static int unshare_fs(unsigned long unsh
}

/*
- * Unshare the mnt_namespace structure if it is being shared
- */
-static int unshare_mnt_namespace(unsigned long unshare_flags,
- struct mnt_namespace **new_nsp, struct fs_struct *new_fs)
-{
- struct mnt_namespace *ns = current->nsproxy->mnt_ns;
-
- if ((unshare_flags & CLONE_NEWNS) && ns) {
- if (!capable(CAP_SYS_ADMIN))
- return -EPERM;
-
- *new_nsp = dup_mnt_ns(current, new_fs ? new_fs : current->fs);
- if (!*new_nsp)
- return -ENOMEM;
- }
-
- return 0;
-}
-
-/*
- * Unsharing of sighand is not supported yet
- */
static int unshare_sighand(unsigned long unshare_flags, struct sighand_struct **new_sighp)
@@ -1592,16 +1572,6 @@ static int unshare_semundo(unsigned long
return 0;
}

#ifdef CONFIG_IPC_NS
-static inline int unshare_ipcs(unsigned long flags, struct ipc_namespace **ns)
-{
- if (flags & CLONE_NEWIPC)
- return -EINVAL;
-
- return 0;
-}
#endif

/*
- * unshare allows a process to 'unshare' part of the process
- * context which was originally shared using clone. copy_*
@@ -1614,14 +1584,11 @@ asmlinkage long sys_unshare(unsigned lon
{
int err = 0;
struct fs_struct *fs, *new_fs = NULL;

```

```

- struct mnt_namespace *ns, *new_ns = NULL;
  struct sighand_struct *new_sigh = NULL;
  struct mm_struct *mm, *new_mm = NULL, *active_mm = NULL;
  struct files_struct *fd, *new_fd = NULL;
  struct sem_undo_list *new_ulist = NULL;
  struct nsproxy *new_nsproxy = NULL, *old_nsproxy = NULL;
- struct uts_namespace *uts, *new_uts = NULL;
- struct ipc_namespace *ipc, *new_ipc = NULL;

```

```

  check_unshare_flags(&unshare_flags);

```

```

@@ -1636,36 +1603,24 @@ asmlinkage long sys_unshare(unsigned long
  goto bad_unshare_out;
  if ((err = unshare_fs(unshare_flags, &new_fs)))
    goto bad_unshare_cleanup_thread;
- if ((err = unshare_mnt_namespace(unshare_flags, &new_ns, new_fs)))
- goto bad_unshare_cleanup_fs;
  if ((err = unshare_sighand(unshare_flags, &new_sigh)))
- goto bad_unshare_cleanup_ns;
+ goto bad_unshare_cleanup_fs;
  if ((err = unshare_vm(unshare_flags, &new_mm)))
    goto bad_unshare_cleanup_sigh;
  if ((err = unshare_fd(unshare_flags, &new_fd)))
    goto bad_unshare_cleanup_vm;
  if ((err = unshare_semundo(unshare_flags, &new_ulist)))
    goto bad_unshare_cleanup_fd;
- if ((err = unshare_utsname(unshare_flags, &new_uts)))
+ if ((err = unshare_nsproxy_namespaces(unshare_flags, &new_nsproxy,
+ new_fs)))
    goto bad_unshare_cleanup_semundo;
- if ((err = unshare_ipcs(unshare_flags, &new_ipc)))
- goto bad_unshare_cleanup_uts;

- if (new_ns || new_uts || new_ipc) {
- old_nsproxy = current->nsproxy;
- new_nsproxy = dup_namespaces(old_nsproxy);
- if (!new_nsproxy) {
- err = -ENOMEM;
- goto bad_unshare_cleanup_ipc;
- }
- }
-
- if (new_fs || new_ns || new_mm || new_fd || new_ulist ||
- new_uts || new_ipc) {
+ if (new_fs || new_mm || new_fd || new_ulist || new_nsproxy) {

```

```

  task_lock(current);

```

```

    if (new_nsproxy) {
+   old_nsproxy = current->nsproxy;
    current->nsproxy = new_nsproxy;
    new_nsproxy = old_nsproxy;
    }
@@ -1676,12 +1631,6 @@ asmlinkage long sys_unshare(unsigned lon
    new_fs = fs;
    }

-   if (new_ns) {
-   ns = current->nsproxy->mnt_ns;
-   current->nsproxy->mnt_ns = new_ns;
-   new_ns = ns;
-   }
-
    if (new_mm) {
    mm = current->mm;
    active_mm = current->active_mm;
@@ -1697,32 +1646,12 @@ asmlinkage long sys_unshare(unsigned lon
    new_fd = fd;
    }

-   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_cleanup_ipc:
-   if (new_ipc)
-   put_ipc_ns(new_ipc);
-
-bad_unshare_cleanup_uts:
-   if (new_uts)
-   put_uts_ns(new_uts);
-

```

```
bad_unshare_cleanup_semundo:
bad_unshare_cleanup_fd:
    if (new_fd)
@@ -1737,10 +1666,6 @@ bad_unshare_cleanup_sigh:
    if (atomic_dec_and_test(&new_sigh->count))
        kmem_cache_free(sighand_cache, new_sigh);

-bad_unshare_cleanup_ns:
- if (new_ns)
- put_mnt_ns(new_ns);
-
bad_unshare_cleanup_fs:
    if (new_fs)
        put_fs_struct(new_fs);
```

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