Subject: - merge-sys_clone-sys_unshare-nsproxy-and-namespace.patch removed from -mm tree
Posted by akpm on Wed, 09 May 2007 02:45:35 GMT
View Forum Message <> Reply to Message

The patch titled
    Merge sys_clone()/sys_unshare() nsproxy and namespace handling
has been removed from the -mm tree. Its filename was
    merge-sys_clone-sys_unshare-nsproxy-and-namespace.patch

This patch was dropped because it was merged into mainline or a subsystem tree

---
Subject: Merge sys_clone()/sys_unshare() nsproxy and namespace handling
From: Badari Pulavarty <pbadari@us.ibm.com>

sys_clone() and sys_unshare() both makes copies of nsproxy and its associated
namespaces. But they have different code paths.

This patch merges all the nsproxy and its associated namespace copy/clone
handling (as much as possible). Posted on container list earlier for
feedback.

- 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 return a new copy of namespace
  instead of attaching it to task->nsproxy.

- Moved the CAP_SYS_ADMIN checks out of copy_*_ns() routines.

- Removed unnessary !ns checks from copy_*_ns() and added BUG_ON()
  just incase.

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

[akpm@osdl.org: cleanups, warning fix]
[clg@fr.ibm.com: remove dup_namespaces() declaration]
[serue@us.ibm.com: fix CONFIG_IPC_NS=n, clone(CLONE_NEWIPC) retval]
[akpm@linux-foundation.org: fix build with CONFIG_SYSVIPC=n]
Signed-off-by: Badari Pulavarty <pbadari@us.ibm.com>
Signed-off-by: Serge Hallyn <serue@us.ibm.com>
Cc: Cedric Le Goater <clg@fr.ibm.com>
Cc: "Eric W. Biederman" <ebiederm@xmission.com>
Cc: <containers@lists.osdl.org>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

Cc: Oleg Nesterov <oleg@tv-sign.ru>
Signed-off-by: Andrew Morton <akpm@linux-foundation.org>

```
 fs/namespace.c                |   30 +-----
 include/linux/ipc.h           |   11 +-
 include/linux/mnt_namespace.h |    5 -
 include/linux/nsproxy.h       |    3
 include/linux/pid_namespace.h |    2
 include/linux/utsname.h       |   19 ----
 ipc/util.c                    |   53 ++----------
 kernel/fork.c                 |   85 +------------------
 kernel/nsproxy.c              |  139 ++++++++++++++++++++-------------
 kernel/pid.c                  |   11 --
 kernel/utsname.c              |   41 ---------
 11 files changed, 131 insertions(+), 268 deletions(-)

diff -puN fs/namespace.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
fs/namespace.c
--- a/fs/namespace.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/fs/namespace.c
@@ -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, *pwdmnt = NULL, *altrootmnt = NULL;
  struct vfsmount *p, *q;
@@ -1509,36 +1508,21 @@ struct mnt_namespace *dup_mnt_ns(struct
  return new_ns;
 }

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

```
+ BUG_ON(!ns);
  get_mnt_ns(ns);

  if (!(flags & CLONE_NEWNS))
-  return 0;
+  return ns;

- if (!capable(CAP_SYS_ADMIN)) {
-  err = -EPERM;
-  goto out;
- }
+ new_ns = dup_mnt_ns(ns, new_fs);

- new_ns = dup_mnt_ns(tsk, tsk->fs);
- if (!new_ns) {
-  err = -ENOMEM;
-  goto out;
- }
-
- tsk->nsproxy->mnt_ns = new_ns;
-
-out:
  put_mnt_ns(ns);
- return err;
+ return new_ns;
 }

 asmlinkage long sys_mount(char __user * dev_name, char __user * dir_name,
diff -puN include/linux/ipc.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
include/linux/ipc.h
--- a/include/linux/ipc.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/include/linux/ipc.h
@@ -92,16 +92,19 @@ extern struct ipc_namespace init_ipc_ns;

 #ifdef CONFIG_SYSVIPC
 #define INIT_IPC_NS(ns)  .ns  = &init_ipc_ns,
-extern int copy_ipcs(unsigned long flags, struct task_struct *tsk);
+extern struct ipc_namespace *copy_ipcs(unsigned long flags,
+       struct ipc_namespace *ns);
 #else
 #define INIT_IPC_NS(ns)
-static inline int copy_ipcs(unsigned long flags, struct task_struct *tsk)
-{ return 0; }
+static inline struct ipc_namespace *copy_ipcs(unsigned long flags,
+       struct ipc_namespace *ns)
+{
+ return ns;
+}
```

```
 #endif

 #ifdef CONFIG_IPC_NS
 extern void free_ipc_ns(struct kref *kref);
-extern int unshare_ipcs(unsigned long flags, struct ipc_namespace **ns);
 #endif

 static inline struct ipc_namespace *get_ipc_ns(struct ipc_namespace *ns)
diff -puN
include/linux/mnt_namespace.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
include/linux/mnt_namespace.h
--- a/include/linux/mnt_namespace.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/include/linux/mnt_namespace.h
@@ -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 struct mnt_namespace *copy_mnt_ns(int, struct mnt_namespace *,
   struct fs_struct *);
+extern void __put_mnt_ns(struct mnt_namespace *ns);

 static inline void put_mnt_ns(struct mnt_namespace *ns)
 {
diff -puN include/linux/nsproxy.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
include/linux/nsproxy.h
--- a/include/linux/nsproxy.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/include/linux/nsproxy.h
@@ -31,10 +31,11 @@ struct nsproxy {
 };
 extern struct nsproxy init_nsproxy;

-struct nsproxy *dup_namespaces(struct nsproxy *orig);
 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)
 {
diff -puN include/linux/pid_namespace.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
include/linux/pid_namespace.h
--- a/include/linux/pid_namespace.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/include/linux/pid_namespace.h
@@ -29,7 +29,7 @@ static inline void get_pid_ns(struct pid
```

```
  kref_get(&ns->kref);
 }

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

 static inline void put_pid_ns(struct pid_namespace *ns)
diff -puN include/linux/utsname.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
include/linux/utsname.h
--- a/include/linux/utsname.h~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/include/linux/utsname.h
@@ -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 struct uts_namespace *copy_utsname(int flags, struct uts_namespace *ns);
 extern void free_uts_ns(struct kref *kref);

 static inline void put_uts_ns(struct uts_namespace *ns)
@@ -59,21 +57,12 @@ 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)
+static inline struct uts_namespace *copy_utsname(int flags,
+       struct uts_namespace *ns)
 {
- if (unshare_flags & CLONE_NEWUTS)
-  return -EINVAL;
-
- return 0;
+ return ns;
 }

-static inline int copy_utsname(int flags, struct task_struct *tsk)
-{
- if (flags & CLONE_NEWUTS)
-  return -EINVAL;
- return 0;
-}
 static inline void put_uts_ns(struct uts_namespace *ns)
 {
 }
```

```
diff -puN ipc/util.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace ipc/util.c
--- a/ipc/util.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/ipc/util.c
@@ -85,53 +85,20 @@ err_mem:
  return ERR_PTR(err);
 }

-int unshare_ipcs(unsigned long unshare_flags, struct ipc_namespace **new_ipc)
+struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
 {
- 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;

- if (!old_ns)
-  return 0;
-
- get_ipc_ns(old_ns);
+ BUG_ON(!ns);
+ get_ipc_ns(ns);

  if (!(flags & CLONE_NEWIPC))
-  return 0;
-
- if (!capable(CAP_SYS_ADMIN)) {
-  err = -EPERM;
-  goto out;
- }
+  return ns;
```

```
- new_ns = clone_ipc_ns(old_ns);
- if (!new_ns) {
-  err = -ENOMEM;
-  goto out;
- }
+ new_ns = clone_ipc_ns(ns);

- tsk->nsproxy->ipc_ns = new_ns;
-out:
- put_ipc_ns(old_ns);
- return err;
+ put_ipc_ns(ns);
+ return new_ns;
 }

 void free_ipc_ns(struct kref *kref)
@@ -145,11 +112,11 @@ void free_ipc_ns(struct kref *kref)
  kfree(ns);
 }
 #else
-int copy_ipcs(unsigned long flags, struct task_struct *tsk)
+struct ipc_namespace *copy_ipcs(unsigned long flags, struct ipc_namespace *ns)
 {
  if (flags & CLONE_NEWIPC)
-  return -EINVAL;
- return 0;
+  return ERR_PTR(-EINVAL);
+ return ns;
 }
 #endif

diff -puN kernel/fork.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace kernel/fork.c
--- a/kernel/fork.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/kernel/fork.c
@@ -1516,26 +1516,6 @@ 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)
@@ -1593,16 +1573,6 @@ static int unshare_semundo(unsigned long
  return 0;
 }

-#ifndef 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_*
@@ -1615,14 +1585,11 @@ 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);

@@ -1637,36 +1604,24 @@ asmlinkage long sys_unshare(unsigned lon
   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;
   }
@@ -1677,12 +1632,6 @@ 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;
@@ -1698,32 +1647,12 @@ 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)
@@ -1738,10 +1667,6 @@ bad_unshare_cleanup_sigh:
    if (atomic_dec_and_test(&new_sigh->count))
     kmem_cache_free(sighand_cachep, 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);
```

```
diff -puN kernel/nsproxy.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
kernel/nsproxy.c
--- a/kernel/nsproxy.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/kernel/nsproxy.c
@@ -38,10 +38,8 @@ 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)
+static inline struct nsproxy *clone_nsproxy(struct nsproxy *orig)
 {
  struct nsproxy *ns;

@@ -52,26 +50,49 @@ 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;

- 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_nsproxy(tsk->nsproxy);
+ if (!new_nsp)
+  return ERR_PTR(-ENOMEM);
```

```
- return ns;
+ new_nsp->mnt_ns = copy_mnt_ns(flags, tsk->nsproxy->mnt_ns, new_fs);
+ if (IS_ERR(new_nsp->mnt_ns))
+  goto out_ns;
+
+ new_nsp->uts_ns = copy_utsname(flags, tsk->nsproxy->uts_ns);
+ if (IS_ERR(new_nsp->uts_ns))
+  goto out_uts;
+
+ new_nsp->ipc_ns = copy_ipcs(flags, tsk->nsproxy->ipc_ns);
+ if (IS_ERR(new_nsp->ipc_ns))
+  goto out_ipc;
+
+ new_nsp->pid_ns = copy_pid_ns(flags, tsk->nsproxy->pid_ns);
+ if (IS_ERR(new_nsp->pid_ns))
+  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(-ENOMEM);
 }

 /*
@@ -92,47 +113,21 @@ 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;
+ if (!capable(CAP_SYS_ADMIN)) {
+  err = -EPERM;
   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;
+ 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;
 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 +142,41 @@ 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 sucess, 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;
+}
diff -puN kernel/pid.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace kernel/pid.c
--- a/kernel/pid.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/kernel/pid.c
@@ -360,16 +360,11 @@ struct pid *find_ge_pid(int nr)
 }
 EXPORT_SYMBOL_GPL(find_get_pid);

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

```
-
- if (!old_ns)
-  return 0;
-
+ BUG_ON(!old_ns);
  get_pid_ns(old_ns);
- return err;
+ return old_ns;
 }

 void free_pid_ns(struct kref *kref)
diff -puN kernel/utsname.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
kernel/utsname.c
--- a/kernel/utsname.c~merge-sys_clone-sys_unshare-nsproxy-and-namespace
+++ a/kernel/utsname.c
@@ -32,58 +32,25 @@ 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)
+struct uts_namespace *copy_utsname(int flags, struct uts_namespace *old_ns)
 {
- struct uts_namespace *old_ns = tsk->nsproxy->uts_ns;
  struct uts_namespace *new_ns;
- int err = 0;
```

```
-
- if (!old_ns)
-  return 0;

+ BUG_ON(!old_ns);
  get_uts_ns(old_ns);

  if (!(flags & CLONE_NEWUTS))
-  return 0;
-
- if (!capable(CAP_SYS_ADMIN)) {
-  err = -EPERM;
-  goto out;
- }
+  return old_ns;

  new_ns = clone_uts_ns(old_ns);
- if (!new_ns) {
-  err = -ENOMEM;
-  goto out;
- }
- tsk->nsproxy->uts_ns = new_ns;

-out:
  put_uts_ns(old_ns);
- return err;
+ return new_ns;
 }

 void free_uts_ns(struct kref *kref)
```

_

Patches currently in -mm which might be from pbadari@us.ibm.com are

origin.patch

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