
Subject: [patch -mm 17/17] user namespace: add unshare
Posted by [Cedric Le Goater](#) on Tue, 05 Dec 2006 10:28:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

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

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

```
include/linux/user_namespace.h | 11 +++++++
kernel/nsproxy.c                | 26 ++++++++
kernel/user_namespace.c         | 57 ++++++++
3 files changed, 91 insertions(+), 3 deletions(-)
```

Index: 2.6.19-rc6-mm2/kernel/nsproxy.c

=====

```
--- 2.6.19-rc6-mm2.orig/kernel/nsproxy.c
+++ 2.6.19-rc6-mm2/kernel/nsproxy.c
@@ -335,6 +335,12 @@ static int switch_ns(int id, unsigned lo
     new_ns->net_ns = ns->net_ns;
 }

+ if (flags & NS_USER) {
+     get_user_ns(ns->user_ns);
+     put_user_ns(new_ns->user_ns);
+     new_ns->user_ns = ns->user_ns;
+ }
+
+ out_ns:
+     put_nsproxy(ns);
+ }
@@ -453,6 +459,7 @@ asmlinkage long sys_unshare_ns(unsigned
     struct ipc_namespace *ipc, *new_ipc = NULL;
     struct pid_namespace *pid, *new_pid = NULL;
     struct net_namespace *net, *new_net = NULL;
+ struct user_namespace *user, *new_user = NULL;
     unsigned long unshare_flags = 0;

     /* Return -EINVAL for all unsupported flags */
@@ -484,17 +491,20 @@ asmlinkage long sys_unshare_ns(unsigned
     goto bad_unshare_ns_cleanup_ipc;
     if ((err = unshare_net_ns(unshare_ns_flags, &new_net)))
         goto bad_unshare_ns_cleanup_pid;
+ if ((err = unshare_user_ns(unshare_ns_flags, &new_user)))
+     goto bad_unshare_ns_cleanup_net;

- if (new_mnt || new_uts || new_ipc || new_pid || new_net) {
+ if (new_mnt || new_uts || new_ipc || new_pid || new_net || new_user) {
     old_nsproxy = current->nsproxy;
```

```

    new_nsproxy = dup_namespaces(old_nsproxy);
    if (!new_nsproxy) {
        err = -ENOMEM;
-   goto bad_unshare_ns_cleanup_net;
+   goto bad_unshare_ns_cleanup_user;
    }
}

- if (new_fs || new_mnt || new_uts || new_ipc || new_pid || new_net) {
+ if (new_fs || new_mnt || new_uts || new_ipc || new_pid || new_net ||
+     new_user) {

    task_lock(current);

@@ -539,12 +549,22 @@ asmlinkage long sys_unshare_ns(unsigned
    new_net = net;
}

+ if (new_user) {
+     user = current->nsproxy->user_ns;
+     current->nsproxy->user_ns = new_user;
+     new_user = user;
+ }
+
    task_unlock(current);
}

if (new_nsproxy)
    put_nsproxy(new_nsproxy);

+bad_unshare_ns_cleanup_user:
+ if (new_user)
+     put_user_ns(new_user);
+
bad_unshare_ns_cleanup_net:
    if (new_net)
        put_net_ns(new_net);
Index: 2.6.19-rc6-mm2/include/linux/user_namespace.h
=====
--- 2.6.19-rc6-mm2.orig/include/linux/user_namespace.h
+++ 2.6.19-rc6-mm2/include/linux/user_namespace.h
@@ -22,6 +22,8 @@ static inline void get_user_ns(struct us
    kref_get(&ns->kref);
}

+extern int unshare_user_ns(unsigned long unshare_flags,
+    struct user_namespace **new_user);
extern int copy_user_ns(int flags, struct task_struct *tsk);

```

```
extern void free_user_ns(struct kref *kref);
```

```
@@ -36,6 +38,15 @@ static inline void get_user_ns(struct us
{
}
```

```
+static inline int unshare_user_ns(unsigned long unshare_flags,
+ struct user_namespace **new_user)
+{
+ if (unshare_flags & NS_USER)
+ return -EINVAL;
+
+ return 0;
+}
+
static inline int copy_user_ns(int flags, struct task_struct *tsk)
{
return 0;
```

```
Index: 2.6.19-rc6-mm2/kernel/user_namespace.c
```

```
=====
```

```
--- 2.6.19-rc6-mm2.orig/kernel/user_namespace.c
```

```
+++ 2.6.19-rc6-mm2/kernel/user_namespace.c
```

```
@@ -21,6 +21,63 @@ EXPORT_SYMBOL_GPL(init_user_ns);
```

```
#ifdef CONFIG_USER_NS
```

```
+/
+ * Clone a new ns copying an original user ns, setting refcount to 1
+ * @old_ns: namespace to clone
+ * Return NULL on error (failure to kmalloc), new ns otherwise
+ */
+static struct user_namespace *clone_user_ns(struct user_namespace *old_ns)
+{
+ struct user_namespace *ns;
+ struct user_struct *new_user;
+ int n;
+
+ ns = kmalloc(sizeof(struct user_namespace), GFP_KERNEL);
+ if (!ns)
+ return NULL;
+
+ kref_init(&ns->kref);
+
+ for(n = 0; n < UIDHASH_SZ; ++n)
+ INIT_LIST_HEAD(ns->uidhash_table + n);
+
+ /* Insert new root user. */
+ ns->root_user = alloc_uid(ns, 0);
```

```

+ if (!ns->root_user) {
+   kfree(ns);
+   return NULL;
+ }
+
+ /* Reset current->user with a new one */
+ new_user = alloc_uid(ns, current->uid);
+ if (!new_user) {
+   free_uid(ns->root_user);
+   kfree(ns);
+   return NULL;
+ }
+
+ switch_uid(new_user);
+ return ns;
+}
+
+/*
+ * unshare the current process' user namespace.
+ */
+int unshare_user_ns(unsigned long unshare_flags,
+   struct user_namespace **new_user)
+{
+   if (unshare_flags & NS_USER) {
+     if (!capable(CAP_SYS_ADMIN))
+       return -EPERM;
+
+     *new_user = clone_user_ns(current->nsproxy->user_ns);
+     if (!*new_user)
+       return -ENOMEM;
+   }
+
+   return 0;
+}
+
+int copy_user_ns(int flags, struct task_struct *tsk)
+{
+   struct user_namespace *old_ns = tsk->nsproxy->user_ns;
+
+   --

```

Containers mailing list

Containers@lists.osdl.org

<https://lists.osdl.org/mailman/listinfo/containers>
