Subject: [PATCH 2.6.24-rc8-mm1 02/15] IPC/shared memory: introduce shmctl_down

Posted by Pierre Peiffer on Tue, 29 Jan 2008 16:02:31 GMT

View Forum Message <> Reply to Message

From: Pierre Peiffer <pierre.peiffer@bull.net>

Currently, the way the different commands are handled in sys_shmctl
introduces some duplicated code.
This patch introduces the shmctl_down function to handle all the commands
requiring the rwmutex to be taken in write mode (ie IPC_SET and IPC_RMID
for now). It is the equivalent function of semctl_down for shared
memory.

This removes some duplicated code for handling these both commands
and harmonizes the way they are handled among all IPCs.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>
Acked-by: Serge Hallyn <serue@us.ibm.com>

```
 ipc/shm.c |  160 +++++++++++++++++++++++++++-----------------------------------
 1 file changed, 72 insertions(+), 88 deletions(-)

Index: b/ipc/shm.c
===================================================================
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -625,10 +625,78 @@ static void shm_get_stat(struct ipc_name
  }
 }

-asmlinkage long sys_shmctl (int shmid, int cmd, struct shmid_ds __user *buf)
+/*
+ * This function handles some shmctl commands which require the rw_mutex
+ * to be held in write mode.
+ * NOTE: no locks must be held, the rw_mutex is taken inside this function.
+ */
+static int shmctl_down(struct ipc_namespace *ns, int shmid, int cmd,
+        struct shmid_ds __user *buf, int version)
 {
+ struct kern_ipc_perm *ipcp;
  struct shm_setbuf setbuf;
  struct shmid_kernel *shp;
+ int err;
+
+ if (cmd == IPC_SET) {
+  if (copy_shmid_from_user(&setbuf, buf, version))
```

```
+   return -EFAULT;
+ }
+
+ down_write(&shm_ids(ns).rw_mutex);
+ shp = shm_lock_check_down(ns, shmid);
+ if (IS_ERR(shp)) {
+  err = PTR_ERR(shp);
+  goto out_up;
+ }
+
+ ipcp = &shp->shm_perm;
+
+ err = audit_ipc_obj(ipcp);
+ if (err)
+  goto out_unlock;
+
+ if (cmd == IPC_SET) {
+  err = audit_ipc_set_perm(0, setbuf.uid,
+       setbuf.gid, setbuf.mode);
+  if (err)
+   goto out_unlock;
+ }
+
+ if (current->euid != ipcp->uid &&
+     current->euid != ipcp->cuid &&
+     !capable(CAP_SYS_ADMIN)) {
+  err = -EPERM;
+  goto out_unlock;
+ }
+
+ err = security_shm_shmctl(shp, cmd);
+ if (err)
+  goto out_unlock;
+ switch (cmd) {
+ case IPC_RMID:
+  do_shm_rmid(ns, ipcp);
+  goto out_up;
+ case IPC_SET:
+  ipcp->uid = setbuf.uid;
+  ipcp->gid = setbuf.gid;
+  ipcp->mode = (ipcp->mode & ~S_IRWXUGO)
+   | (setbuf.mode & S_IRWXUGO);
+  shp->shm_ctim = get_seconds();
+  break;
+ default:
+  err = -EINVAL;
+ }
+out_unlock:
```

```
+ shm_unlock(shp);
+out_up:
+ up_write(&shm_ids(ns).rw_mutex);
+ return err;
+}
+
+asmlinkage long sys_shmctl(int shmid, int cmd, struct shmid_ds __user *buf)
+{
+ struct shmid_kernel *shp;
  int err, version;
  struct ipc_namespace *ns;

@@ -784,97 +852,13 @@ asmlinkage long sys_shmctl (int shmid, i
   goto out;
  }
  case IPC_RMID:
- {
-  /*
-   * We cannot simply remove the file. The SVID states
-   * that the block remains until the last person
-   * detaches from it, then is deleted. A shmat() on
-   * an RMID segment is legal in older Linux and if
-   * we change it apps break...
-   *
-   * Instead we set a destroyed flag, and then blow
-   * the name away when the usage hits zero.
-   */
-  down_write(&shm_ids(ns).rw_mutex);
-  shp = shm_lock_check_down(ns, shmid);
-  if (IS_ERR(shp)) {
-   err = PTR_ERR(shp);
-   goto out_up;
-  }
-
-  err = audit_ipc_obj(&(shp->shm_perm));
-  if (err)
-   goto out_unlock_up;
-
-  if (current->euid != shp->shm_perm.uid &&
-      current->euid != shp->shm_perm.cuid &&
-      !capable(CAP_SYS_ADMIN)) {
-   err=-EPERM;
-   goto out_unlock_up;
-  }
-
-  err = security_shm_shmctl(shp, cmd);
-  if (err)
-   goto out_unlock_up;
```

```
-
-  do_shm_rmid(ns, &shp->shm_perm);
-  up_write(&shm_ids(ns).rw_mutex);
-  goto out;
- }
-
   case IPC_SET:
- {
-  if (!buf) {
-   err = -EFAULT;
-   goto out;
-  }
-
-  if (copy_shmid_from_user (&setbuf, buf, version)) {
-   err = -EFAULT;
-   goto out;
-  }
-  down_write(&shm_ids(ns).rw_mutex);
-  shp = shm_lock_check_down(ns, shmid);
-  if (IS_ERR(shp)) {
-   err = PTR_ERR(shp);
-   goto out_up;
-  }
-  err = audit_ipc_obj(&(shp->shm_perm));
-  if (err)
-   goto out_unlock_up;
-  err = audit_ipc_set_perm(0, setbuf.uid, setbuf.gid, setbuf.mode);
-  if (err)
-   goto out_unlock_up;
-  err=-EPERM;
-  if (current->euid != shp->shm_perm.uid &&
-      current->euid != shp->shm_perm.cuid &&
-      !capable(CAP_SYS_ADMIN)) {
-   goto out_unlock_up;
-  }
-
-  err = security_shm_shmctl(shp, cmd);
-  if (err)
-   goto out_unlock_up;
-
-  shp->shm_perm.uid = setbuf.uid;
-  shp->shm_perm.gid = setbuf.gid;
-  shp->shm_perm.mode = (shp->shm_perm.mode & ~S_IRWXUGO)
-   | (setbuf.mode & S_IRWXUGO);
-  shp->shm_ctim = get_seconds();
-  break;
- }
-
```

```
+  err = shmctl_down(ns, shmid, cmd, buf, version);
+  return err;
  default:
-  err = -EINVAL;
-  goto out;
+  return -EINVAL;
  }

- err = 0;
-out_unlock_up:
- shm_unlock(shp);
-out_up:
- up_write(&shm_ids(ns).rw_mutex);
- goto out;
 out_unlock:
  shm_unlock(shp);
 out:
```

--
Pierre Peiffer

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