Subject: [RFC][PATCH 1/2] System V IPC: new kernel API to change an ID
Posted by Pierre Peiffer on Fri, 28 Sep 2007 14:58:37 GMT
View Forum Message <> Reply to Message

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

This patch provides three new API for changing the ID of an existing
System V IPCs.

These APIs are:

```
 long msg_mvid(struct ipc_namespace *ns, int id, int newid);
 long sem_mvid(struct ipc_namespace *ns, int id, int newid);
 long shm_mvid(struct ipc_namespace *ns, int id, int newid);
```

They return 0 or an error code in case of failure.

They may be useful for setting a specific ID for an IPC when preparing
a restart operation.

To be successful, the following rules must be respected:
- the IPC exists (of course...)
- the new ID must satisfy the ID computation rule.
- the entry (in the kernel internal table of IPCs) corresponding to the new
  ID must be free.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>

```
 include/linux/msg.h |    1 +
 include/linux/sem.h |    1 +
 include/linux/shm.h |    1 +
 ipc/msg.c           |   32 ++++++++++++++++++++++++++++++
 ipc/sem.c           |   32 ++++++++++++++++++++++++++++++
 ipc/shm.c           |   30 ++++++++++++++++++++++++++++
 ipc/util.c          |   60 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 ipc/util.h          |    1 +
 8 files changed, 158 insertions(+), 0 deletions(-)

diff --git a/include/linux/msg.h b/include/linux/msg.h
index f1b6074..5a1db95 100644
--- a/include/linux/msg.h
+++ b/include/linux/msg.h
@@ -97,6 +97,7 @@ extern long do_msgsnd(int msqid, long mtype, void __user *mtext,
    size_t msgsz, int msgflg);
 extern long do_msgrcv(int msqid, long *pmtype, void __user *mtext,
    size_t msgsz, long msgtyp, int msgflg);
+long msg_mvid(struct ipc_namespace *ns, int id, int newid);
```

```
 #endif /* __KERNEL__ */

diff --git a/include/linux/sem.h b/include/linux/sem.h
index 9aaffb0..b5989fb 100644
--- a/include/linux/sem.h
+++ b/include/linux/sem.h
@@ -142,6 +142,7 @@ struct sysv_sem {

 extern int copy_semundo(unsigned long clone_flags, struct task_struct *tsk);
 extern void exit_sem(struct task_struct *tsk);
+long sem_mvid(struct ipc_namespace *ns, int id, int newid);

 #else
 static inline int copy_semundo(unsigned long clone_flags, struct task_struct *tsk)
diff --git a/include/linux/shm.h b/include/linux/shm.h
index ad2e3af..f4ae995 100644
--- a/include/linux/shm.h
+++ b/include/linux/shm.h
@@ -97,6 +97,7 @@ struct shmid_kernel /* private to the kernel */
 #ifdef CONFIG_SYSVIPC
 long do_shmat(int shmid, char __user *shmaddr, int shmflg, unsigned long *addr);
 extern int is_file_shm_hugepages(struct file *file);
+long shm_mvid(struct ipc_namespace *ns, int id, int newid);
 #else
 static inline long do_shmat(int shmid, char __user *shmaddr,
     int shmflg, unsigned long *addr)
diff --git a/ipc/msg.c b/ipc/msg.c
index a03fcb5..d9d4093 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -382,6 +382,38 @@ copy_msqid_from_user(struct msq_setbuf *out, void __user *buf, int
version)
  }
 }

+long msg_mvid(struct ipc_namespace *ns, int id, int newid)
+{
+ long err;
+ struct msg_queue *msq;
+
+ mutex_lock(&msg_ids(ns).mutex);
+ msq = msg_lock(ns, id);
+
+ err = -EINVAL;
+ if (msq == NULL)
+   goto out_up;
+
+ err = msg_checkid(ns, msq, id);
```

```
+ if (err)
+  goto out_unlock_up;
+
+ err = ipc_mvid(&msg_ids(ns), id,
+         newid, ns->msg_ctlmni);
+
+ if (err)
+  goto out_unlock_up;
+
+ msq->q_id = newid;
+ msq->q_ctime = get_seconds();
+
+out_unlock_up:
+ msg_unlock(msq);
+out_up:
+ mutex_unlock(&msg_ids(ns).mutex);
+ return err;
+}
+
 asmlinkage long sys_msgctl(int msqid, int cmd, struct msqid_ds __user *buf)
 {
  struct kern_ipc_perm *ipcp;
diff --git a/ipc/sem.c b/ipc/sem.c
index b676fef..606f2e9 100644
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -918,6 +918,38 @@ out_unlock:
  return err;
 }

+long sem_mvid(struct ipc_namespace *ns, int id, int newid)
+{
+ long err;
+ struct sem_array *sma;
+
+ mutex_lock(&sem_ids(ns).mutex);
+ sma = sem_lock(ns, id);
+
+ err = -EINVAL;
+ if (sma == NULL)
+   goto out_up;
+
+ err = sem_checkid(ns, sma, id);
+ if (err)
+  goto out_unlock_up;
+
+ err = ipc_mvid(&sem_ids(ns), id,
+         newid, ns->sc_semmni);
```

```
+
+ if (err)
+  goto out_unlock_up;
+
+ sma->sem_id = newid;
+ sma->sem_ctime = get_seconds();
+
+out_unlock_up:
+ sem_unlock(sma);
+out_up:
+ mutex_unlock(&sem_ids(ns).mutex);
+ return err;
+}
+
 asmlinkage long sys_semctl (int semid, int semnum, int cmd, union semun arg)
 {
  int err = -EINVAL;
diff --git a/ipc/shm.c b/ipc/shm.c
index a86a3a5..5f4bca6 100644
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -156,7 +156,37 @@ static inline int shm_addid(struct ipc_namespace *ns, struct
shmid_kernel *shp)
  return ipc_addid(&shm_ids(ns), &shp->shm_perm, ns->shm_ctlmni);
 }

+long shm_mvid(struct ipc_namespace *ns, int id, int newid)
+{
+ long err;
+ struct shmid_kernel *shp;
+
+ mutex_lock(&shm_ids(ns).mutex);
+ shp = shm_lock(ns, id);
+
+ err = -EINVAL;
+ if (shp == NULL)
+   goto out_up;
+
+ err = shm_checkid(ns, shp, id);
+ if (err)
+  goto out_unlock_up;
+
+ err = ipc_mvid(&shm_ids(ns), id,
+         newid, ns->shm_ctlmni);

+ if (err)
+  goto out_unlock_up;
+
```

```
+ shp->id = newid;
+ shp->shm_ctim = get_seconds();
+
+out_unlock_up:
+ shm_unlock(shp);
+out_up:
+ mutex_unlock(&shm_ids(ns).mutex);
+ return err;
+}

 /* This is called by fork, once for every shm attach. */
 static void shm_open(struct vm_area_struct *vma)
diff --git a/ipc/util.c b/ipc/util.c
index 44e5135..4f338d4 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -318,6 +318,66 @@ found:
 }

 /**
+ * ipc_mvid  - move an IPC identifier
+ * @ids: IPC identifier set
+ * @oldid: ID of the IPC permission set to move
+ * @newid: new ID of the IPC permission set to move
+ * @size: new size limit for the id array
+ *
+ * Move an entry in the IPC arrays from the 'oldid' place to the
+ *      'newid' place. The seq number of the entry is updated to match the
+ *      'newid' value.
+ *
+ * Called with the list lock and ipc_ids.mutex held.
+ */
+
+int ipc_mvid(struct ipc_ids *ids, int oldid, int newid, int size)
+{
+ struct kern_ipc_perm *p;
+ int old_lid = oldid % SEQ_MULTIPLIER;
+ int new_lid = newid % SEQ_MULTIPLIER;
+
+ if ((new_lid >= size) ||
+     newid !=  (new_lid + (newid/SEQ_MULTIPLIER)*SEQ_MULTIPLIER))
+  return -ERANGE;
+
+ size = grow_ary(ids, size);
+
+ BUG_ON(old_lid >= ids->entries->size);
+
+ p = ids->entries->p[old_lid];
```

```
+
+ if (!p)
+  return -ENXIO;
+
+ /*

+  * but not the seq number.
+  */
+ if (new_lid != old_lid) {
+
+  if (ids->entries->p[new_lid])
+   return -EBUSY;
+
+  ids->entries->p[new_lid] = p;
+
+  ids->entries->p[old_lid] = NULL;
+
+  if (new_lid > ids->max_id)
+   ids->max_id = new_lid;
+  if (old_lid == ids->max_id) {
+   do {
+    --old_lid;
+   } while (ids->entries->p[old_lid] == NULL);
+   ids->max_id = old_lid;
+  }
+ }
+
+ p->seq = newid/SEQ_MULTIPLIER;
+ return 0;
+}
+
+/**
  * ipc_rmid - remove an IPC identifier
  * @ids: identifier set
  * @id: Identifier to remove
diff --git a/ipc/util.h b/ipc/util.h
index 333e891..886af31 100644
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -59,6 +59,7 @@ int ipc_findkey(struct ipc_ids* ids, key_t key);
 int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size);

 /* must be called with both locks acquired. */
+int ipc_mvid(struct ipc_ids *ids, int oldid, int newid, int size);
 struct kern_ipc_perm* ipc_rmid(struct ipc_ids* ids, int id);

 int ipcperms (struct kern_ipc_perm *ipcp, short flg);
```

--
Pierre

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