Subject: [PATCH 2.6.22-rc5 0/2] Setting/updating the ID of a System V IPC
Posted by Pierre Peiffer on Thu, 21 Jun 2007 14:48:32 GMT
View Forum Message <> Reply to Message

Hi,

The two following patches are an update of the one sent at the beginning of this week.
They introduce the ability of setting or modifying the ID of an existing System V IPC. This may be specially useful in the checkpoint/restart context, when preparing the IPCs ressources before restarting a process.

Following the (good :)) comments, the first patch introduces a kernel API for doing the job from kernel space only.
The second patch adds the possibility of doing the same job from user space, by adding a new IPC_SETID command to the ***ctl functions.

As usual, all comments are welcome :) !

Thanks,

--
Pierre

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

Subject: [PATCH 2.6.22-rc5 1/2] New kernel API for changing an ID of a SystemV IPC
Posted by Pierre Peiffer on Thu, 21 Jun 2007 14:48:33 GMT
View Forum Message <> Reply to Message

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(+)

Index: b/ipc/util.c
===
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -327,6 +327,66 @@ 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;
+
+ /*
+  * The id (n° of the entry in the table entries) may be the same
+  * 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
Index: b/ipc/util.h
===================================================================
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -63,6 +63,7 @@ int ipc_findkey(struct ipc_ids* ids, 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);
Index: b/include/linux/msg.h
===================================================================
--- a/include/linux/msg.h
+++ b/include/linux/msg.h
@@ -97,6 +97,7 @@ extern long do_msgsnd(int msqid, long mt
    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__ */

Index: b/include/linux/sem.h
===================================================================
--- 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)
Index: b/include/linux/shm.h
===================================================================
--- a/include/linux/shm.h
+++ b/include/linux/shm.h
@@ -97,6 +97,7 @@ struct shmid_kernel /* private to the ke
 #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)
Index: b/ipc/msg.c
===================================================================
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -384,6 +384,38 @@ copy_msqid_from_user(struct msq_setbuf *
  }
```

```
 }

+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;
Index: b/ipc/sem.c
===================================================================
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -920,6 +920,38 @@ 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;
Index: b/ipc/shm.c
===================================================================
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -158,7 +158,37 @@ static inline int shm_addid(struct ipc_n
  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)
```

--
Pierre Peiffer

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

---

Subject: [PATCH 2.6.22-rc5 2/2] System V IPC: new IPC_SETID command to modify an ID

Posted by Pierre Peiffer on Thu, 21 Jun 2007 14:48:34 GMT

View Forum Message <> Reply to Message

This patch adds a new IPC_SETID command to the System V IPCs set of commands,
which allows to change the ID of an existing IPC.

This command can be used through the semctl/shmctl/msgctl API, with the new
ID passed as the third argument for msgctl and shmctl (instead of a pointer)
and through the fourth argument for semctl.

To be successful, the following rules must be respected:

- the IPC exists
- the user must be allowed to change the IPC attributes regarding the IPC
  permissions.
- 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/ipc.h      |    9 ++++---
 ipc/msg.c                |   31 ++++++++++++++++++-------
 ipc/sem.c                |   31 ++++++++++++++++++-------
 ipc/shm.c                |   55 +++++++++++++++++++++++++++++++++++++++++--------
 security/selinux/hooks.c |    3 ++
 5 files changed, 100 insertions(+), 29 deletions(-)

Index: b/include/linux/ipc.h
===================================================================
--- a/include/linux/ipc.h
+++ b/include/linux/ipc.h
@@ -35,10 +35,11 @@ struct ipc_perm
  * Control commands used with semctl, msgctl and shmctl
  * see also specific commands in sem.h, msg.h and shm.h
  */
-#define IPC_RMID 0     /* remove resource */
-#define IPC_SET  1     /* set ipc_perm options */
-#define IPC_STAT 2     /* get ipc_perm options */
-#define IPC_INFO 3     /* see ipcs */
+#define IPC_RMID  0     /* remove resource */
+#define IPC_SET   1     /* set ipc_perm options */
+#define IPC_STAT  2     /* get ipc_perm options */
+#define IPC_INFO  3     /* see ipcs */
+#define IPC_SETID 4     /* set ipc ID */

 /*
  * Version flags for semctl, msgctl, and shmctl commands
Index: b/ipc/msg.c
===================================================================
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -81,6 +81,8 @@ static struct ipc_ids init_msg_ids;
 #define msg_buildid(ns, id, seq) \
  ipc_buildid(&msg_ids(ns), id, seq)

+static long msg_chid_nolock(struct ipc_namespace *ns, struct msg_queue *msq,
+       int newid);
 static void freeque (struct ipc_namespace *ns, struct msg_queue *msq, int id);
```

```
 static int newque (struct ipc_namespace *ns, key_t key, int msgflg);
 #ifdef CONFIG_PROC_FS
@@ -384,6 +386,21 @@ copy_msqid_from_user(struct msq_setbuf *
  }
 }

+static long msg_chid_nolock(struct ipc_namespace *ns, struct msg_queue *msq,
+       int newid)
+{
+ long err;
+ err = ipc_mvid(&msg_ids(ns), msq->q_id,
+         newid, ns->msg_ctlmni);
+
+ if (err)
+  return err;
+
+ msq->q_id = newid;
+ msq->q_ctime = get_seconds();
+ return 0;
+}
+
 long msg_mvid(struct ipc_namespace *ns, int id, int newid)
 {
  long err;
@@ -400,14 +417,7 @@ long msg_mvid(struct ipc_namespace *ns,
  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();
+ err = msg_chid_nolock(ns, msq, newid);

 out_unlock_up:
  msg_unlock(msq);
@@ -523,6 +533,7 @@ asmlinkage long sys_msgctl(int msqid, in
   if (copy_msqid_from_user(&setbuf, buf, version))
    return -EFAULT;
   break;
+ case IPC_SETID:
  case IPC_RMID:
   break;
  default:
```

```
@@ -585,6 +596,10 @@ asmlinkage long sys_msgctl(int msqid, in
   msg_unlock(msq);
   break;
  }
+ case IPC_SETID:
+  err = msg_chid_nolock(ns, msq, (int)buf);
+  msg_unlock(msq);
+  break;
  case IPC_RMID:
   freeque(ns, msq, msqid);
   break;
Index: b/ipc/sem.c
===================================================================
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -98,6 +98,8 @@

 static struct ipc_ids init_sem_ids;

+static long sem_chid_nolock(struct ipc_namespace *ns, struct sem_array *sma,
+       int newid);
 static int newary(struct ipc_namespace *, key_t, int, int);
 static void freeary(struct ipc_namespace *ns, struct sem_array *sma, int id);
 #ifdef CONFIG_PROC_FS
@@ -908,6 +910,10 @@ static int semctl_down(struct ipc_namesp
   sem_unlock(sma);
   err = 0;
   break;
+ case IPC_SETID:
+  err = sem_chid_nolock(ns, sma, (int)arg.val);
+  sem_unlock(sma);
+  break;
  default:
   sem_unlock(sma);
   err = -EINVAL;
@@ -920,6 +926,21 @@ out_unlock:
  return err;
 }

+static long sem_chid_nolock(struct ipc_namespace *ns, struct sem_array *sma,
+       int newid)
+{
+ long err;
+ err = ipc_mvid(&sem_ids(ns), sma->sem_id,
+        newid, ns->sc_semmni);
+
+ if (err)
+  return err;
```

```
+
+ sma->sem_id = newid;
+ sma->sem_ctime = get_seconds();
+ return 0;
+}
+
 long sem_mvid(struct ipc_namespace *ns, int id, int newid)
 {
  long err;
@@ -936,14 +957,7 @@ long sem_mvid(struct ipc_namespace *ns,
  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();
+ err = sem_chid_nolock(ns, sma, newid);

 out_unlock_up:
  sem_unlock(sma);
@@ -982,6 +996,7 @@ asmlinkage long sys_semctl (int semid, i
   return err;
  case IPC_RMID:
  case IPC_SET:
+ case IPC_SETID:
   mutex_lock(&sem_ids(ns).mutex);
   err = semctl_down(ns,semid,semnum,cmd,version,arg);
   mutex_unlock(&sem_ids(ns).mutex);
Index: b/security/selinux/hooks.c
===================================================================
--- a/security/selinux/hooks.c
+++ b/security/selinux/hooks.c
@@ -4152,6 +4152,7 @@ static int selinux_msg_queue_msgctl(stru
   perms = MSGQ__GETATTR | MSGQ__ASSOCIATE;
   break;
  case IPC_SET:
+ case IPC_SETID:
   perms = MSGQ__SETATTR;
   break;
  case IPC_RMID:
@@ -4300,6 +4301,7 @@ static int selinux_shm_shmctl(struct shm
   perms = SHM__GETATTR | SHM__ASSOCIATE;
   break;
```

```
  case IPC_SET:
+ case IPC_SETID:
    perms = SHM__SETATTR;
    break;
  case SHM_LOCK:
@@ -4411,6 +4413,7 @@ static int selinux_sem_semctl(struct sem
    perms = SEM__DESTROY;
    break;
  case IPC_SET:
+ case IPC_SETID:
    perms = SEM__SETATTR;
    break;
  case IPC_STAT:
Index: b/ipc/shm.c
===================================================================
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -70,6 +70,8 @@ static struct ipc_ids init_shm_ids;

 static int newseg (struct ipc_namespace *ns, key_t key,
    int shmflg, size_t size);
+static long shm_chid_nolock(struct ipc_namespace *ns, struct shmid_kernel *shp,
+       int newid);
 static void shm_open(struct vm_area_struct *vma);
 static void shm_close(struct vm_area_struct *vma);
 static void shm_destroy (struct ipc_namespace *ns, struct shmid_kernel *shp);
@@ -158,6 +160,21 @@ static inline int shm_addid(struct ipc_n
  return ipc_addid(&shm_ids(ns), &shp->shm_perm, ns->shm_ctlmni);
 }

+static long shm_chid_nolock(struct ipc_namespace *ns, struct shmid_kernel *shp,
+       int newid)
+{
+ long err;
+ err = ipc_mvid(&shm_ids(ns), shp->id,
+         newid, ns->shm_ctlmni);
+
+ if (err)
+  return err;
+
+ shp->id = newid;
+ shp->shm_ctim = get_seconds();
+ return 0;
+}
+
 long shm_mvid(struct ipc_namespace *ns, int id, int newid)
 {
  long err;
```

```
@@ -174,14 +191,7 @@ long shm_mvid(struct ipc_namespace *ns,
  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();
+ err = shm_chid_nolock(ns, shp, newid);

 out_unlock_up:
  shm_unlock(shp);
@@ -839,12 +849,39 @@ asmlinkage long sys_shmctl (int shmid, i
   break;
  }

+ case IPC_SETID:
+ {
+  mutex_lock(&shm_ids(ns).mutex);
+  shp = shm_lock(ns, shmid);
+  err = -EINVAL;
+  if(shp == NULL)
+   goto out_up;
+  err = shm_checkid(ns, shp,shmid);
+  if(err)
+   goto out_unlock_up;
+  err = audit_ipc_obj(&(shp->shm_perm));
+  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;
+
+  err = shm_chid_nolock(ns, shp, (int)buf);
+  break;
+ }
+
```

```
  default:
   err = -EINVAL;
   goto out;
  }

- err = 0;
 out_unlock_up:
  shm_unlock(shp);
 out_up:
```

--
Pierre Peiffer