Subject: [PATCH v4 2/9] ipc: "use key as id" functionality for resource get system call int
Posted by Stanislav Kinsbursky on Mon, 13 Aug 2012 12:31:42 GMT
View Forum Message <> Reply to Message

This patch introduces new IPC resource get request flag IPC_PRESET, which
should be interpreted as a request to try to allocate IPC slot with number,
starting from value resented by key. IOW, kernel will try
allocate new segment in specified slot. If slot is not emply, them -EEXIST
returned.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Signed-off-by: Cyrill Gorcunov <gorcunov@openvz.org>
---
 include/linux/ipc.h | 1 +
 ipc/msg.c | 1 +
 ipc/sem.c | 1 +
 ipc/shm.c | 1 +
 ipc/util.c | 8 +++++++-
 ipc/util.h | 1 +
 6 files changed, 12 insertions(+), 1 deletions(-)

diff --git a/include/linux/ipc.h b/include/linux/ipc.h
index 30e8161..d7e5632 100644
--- a/include/linux/ipc.h
+++ b/include/linux/ipc.h
@@ -24,6 +24,7 @@ struct ipc_perm
 #define IPC_CREAT 00001000   /* create if key is nonexistent */
 #define IPC_EXCL   00002000   /* fail if key exists */
 #define IPC_NOWAIT 00004000   /* return error on wait */
+#define IPC_PRESET 00040000   /* use key as id */

 /* these fields are used by the DIPC package so the kernel as standard
    should avoid using them if possible */
diff --git a/ipc/msg.c b/ipc/msg.c
index f3bfbb8..70939a0 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -190,6 +190,7 @@ static int newque(struct ipc_namespace *ns, struct ipc_params *params)

  msq->q_perm.mode = msgflg & S_IRWXUGO;
  msq->q_perm.key = key;
+ msq->q_perm.id = (msgflg & IPC_PRESET) ? key : 0;

  msq->q_perm.security = NULL;
  retval = security_msg_queue_alloc(msq);
diff --git a/ipc/sem.c b/ipc/sem.c
index 5215a81..845c912 100644

```
--- a/ipc/sem.c
+++ b/ipc/sem.c
@@ -306,6 +306,7 @@ static int newary(struct ipc_namespace *ns, struct ipc_params *params)

  sma->sem_perm.mode = (semflg & S_IRWXUGO);
  sma->sem_perm.key = key;
+ sma->sem_perm.id = (semflg & IPC_PRESET) ? key : 0;

  sma->sem_perm.security = NULL;
  retval = security_sem_alloc(sma);
diff --git a/ipc/shm.c b/ipc/shm.c
index 41c1285..d4c0a9e 100644
--- a/ipc/shm.c
+++ b/ipc/shm.c
@@ -480,6 +480,7 @@ static int newseg(struct ipc_namespace *ns, struct ipc_params *params)

  shp->shm_perm.key = key;
  shp->shm_perm.mode = (shmflg & S_IRWXUGO);
+ shp->shm_perm.id = (shmflg & IPC_PRESET) ? key : 0;
  shp->mlock_user = NULL;

  shp->shm_perm.security = NULL;
diff --git a/ipc/util.c b/ipc/util.c
index 75261a3..ac9c6a9 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -264,7 +264,8 @@ int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size)
  rcu_read_lock();
  spin_lock(&new->lock);

- err = idr_get_new(&ids->ipcs_idr, new, &id);
+ err = idr_get_new_above(&ids->ipcs_idr, new,
+     ipcid_to_idx(new->id), &id);
  if (err) {
   spin_unlock(&new->lock);
   rcu_read_unlock();
@@ -277,6 +278,11 @@ int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size)
  new->cuid = new->uid = euid;
  new->gid = new->cgid = egid;

+ if (new->id && ipcid_to_idx(new->id) == id) {
+  new->seq = ipcid_to_seq(new->id);
+  return id;
+ }
+
  new->seq = ids->seq++;
  if(ids->seq > ids->seq_max)
   ids->seq = 0;
```

```
diff --git a/ipc/util.h b/ipc/util.h
index 6f5c20b..5f04b02 100644
--- a/ipc/util.h
+++ b/ipc/util.h
@@ -92,6 +92,7 @@ void __init ipc_init_proc_interface(const char *path, const char *header,
 #define IPC_SHM_IDS 2

 #define ipcid_to_idx(id) ((id) % SEQ_MULTIPLIER)
+#define ipcid_to_seq(id) ((id) / SEQ_MULTIPLIER)

 /* must be called with ids->rw_mutex acquired for writing */
 int ipc_addid(struct ipc_ids *, struct kern_ipc_perm *, int);
```