Subject: [RFC PATCH 2/5] use next syscall data to predefine ipc objects ids
Posted by Nadia Derbey on Tue, 08 Jul 2008 11:24:24 GMT
View Forum Message <> Reply to Message

[PATCH 02/05]

This patch uses the value written into the next_syscall_data proc file
as a target id for the next IPC object to be created.
The following syscalls have a new behavior if next_syscall_data is set:
. mssget()
. semget()
. shmget()

Signed-off-by: Nadia Derbey <Nadia.Derbey@bull.net>

```
---
 include/linux/next_syscall_data.h |   17 ++++++++++++++--
 ipc/util.c                        |   39 +++++++++++++++++++++++++++++++--------
 2 files changed, 46 insertions(+), 10 deletions(-)

Index: linux-2.6.26-rc8-mm1/include/linux/next_syscall_data.h
===================================================================
--- linux-2.6.26-rc8-mm1.orig/include/linux/next_syscall_data.h 2008-07-08 09:24:38.000000000
+0200
+++ linux-2.6.26-rc8-mm1/include/linux/next_syscall_data.h 2008-07-08 12:12:39.000000000
+0200
@@ -1,7 +1,10 @@
 /*
  * include/linux/next_syscall_data.h
  *
- * Definitions to support fixed data for next syscall to be called.
+ * Definitions to support fixed data for next syscall to be called. The
+ * following is supported today:
+ *    . object creation with a predefined id
+ *         . for a sysv ipc object
  */

 #ifndef _LINUX_NEXT_SYSCALL_DATA_H
@@ -13,13 +16,23 @@
  * If this structure is pointed to by a task_struct, next syscall to be called
  * by the task will have a non-default behavior.
  * For example, it can be used to pre-set the id of the object to be created
- * by next syscall.
+ * by next syscall. The following syscalls support this feature:
+ *    . msgget(), semget(), shmget()
  */
 struct next_syscall_data {
  int ndata;
```

```
  long data[NDATA];
 };

+/*
+ * Returns true if tsk has some data set in its next_syscall_data, 0 else
+ */
+#define next_data_set(tsk) ((tsk)->nsd    \
+     ? ((tsk)->nsd->ndata ? 1 : 0) \
+     : 0)
+
+#define get_next_data(tsk) ((tsk)->nsd->data[0])
+
 extern ssize_t get_next_syscall_data(struct task_struct *, char *, size_t);
 extern int set_next_syscall_data(struct task_struct *, char *);
 extern void reset_next_syscall_data(struct task_struct *);
Index: linux-2.6.26-rc8-mm1/ipc/util.c
===================================================================
--- linux-2.6.26-rc8-mm1.orig/ipc/util.c 2008-07-08 09:05:09.000000000 +0200
+++ linux-2.6.26-rc8-mm1/ipc/util.c 2008-07-08 12:13:40.000000000 +0200
@@ -266,20 +266,43 @@ int ipc_addid(struct ipc_ids* ids, struc
  if (ids->in_use >= size)
   return -ENOSPC;

- err = idr_get_new(&ids->ipcs_idr, new, &id);
- if (err)
-  return err;
+ if (unlikely(next_data_set(current))) {
+  /* There is a target id specified, try to use it */
+  int next_id = get_next_data(current);
+  int new_lid = next_id % SEQ_MULTIPLIER;
+  unsigned long new_seq = next_id / SEQ_MULTIPLIER;
+
+  reset_next_syscall_data(current);
+
+  if (next_id != (new_lid + (new_seq * SEQ_MULTIPLIER)))
+   return -EINVAL;
+
+  err = idr_get_new_above(&ids->ipcs_idr, new, new_lid, &id);
+  if (err)
+   return err;
+  if (id != new_lid) {
+   idr_remove(&ids->ipcs_idr, id);
+   return -EBUSY;
+  }
+
+  new->id = next_id;
+  new->seq = new_seq;
+ } else {
```

```
+  err = idr_get_new(&ids->ipcs_idr, new, &id);
+  if (err)
+   return err;
+
+  new->seq = ids->seq++;
+  if (ids->seq > ids->seq_max)
+   ids->seq = 0;
+  new->id = ipc_buildid(id, new->seq);
+ }

  ids->in_use++;

  new->cuid = new->uid = current->euid;
  new->gid = new->cgid = current->egid;

- new->seq = ids->seq++;
- if(ids->seq > ids->seq_max)
-  ids->seq = 0;
-
- new->id = ipc_buildid(id, new->seq);
  spin_lock_init(&new->lock);
  new->deleted = 0;
  rcu_read_lock();

--
```