Subject: [PATCH v8 2/5] ipc: add sysctl to specify desired next object id
Posted by Stanislav Kinsbursky on Wed, 24 Oct 2012 15:35:09 GMT
View Forum Message <> Reply to Message

This patch adds 3 new variables and sysctls to tune them (by one "next_id"
variable for messages, semaphores and shared memory respectively).
This variable can be used to set desired id for next allocated IPC object.
By default it's equal to -1 and old behaviour is preserved.
If this variable is non-negative, then desired idr will be extracted from it
and used as a start value to search for free IDR slot.

Notes:
1) this patch doesn't garantee, that new object will have desired id. So it's
up to user space how to handle new object with wrong id.
2) After sucessfull id allocation attempt, "next_id" will be set back to -1
(if it was non-negative).

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
---
 include/linux/ipc_namespace.h |    1 +
 ipc/ipc_sysctl.c              |   28 ++++++++++++++++++++++++++++
 ipc/util.c                    |   16 ++++++++++------
 ipc/util.h                    |    1 +
 4 files changed, 42 insertions(+), 4 deletions(-)

diff --git a/include/linux/ipc_namespace.h b/include/linux/ipc_namespace.h
index 5499c92..8704e40 100644
--- a/include/linux/ipc_namespace.h
+++ b/include/linux/ipc_namespace.h
@@ -24,6 +24,7 @@ struct ipc_ids {
  unsigned short seq_max;
  struct rw_semaphore rw_mutex;
  struct idr ipcs_idr;
+ int next_id;
 };

 struct ipc_namespace {
diff --git a/ipc/ipc_sysctl.c b/ipc/ipc_sysctl.c
index 00fba2b..d06d77a 100644
--- a/ipc/ipc_sysctl.c
+++ b/ipc/ipc_sysctl.c
@@ -158,6 +158,7 @@ static int proc_ipcauto_dointvec_minmax(ctl_table *table, int write,

 static int zero;
 static int one = 1;
+static int int_max = INT_MAX;

 static struct ctl_table ipc_kern_table[] = {

```
 {
@@ -227,6 +228,33 @@ static struct ctl_table ipc_kern_table[] = {
   .extra1  = &zero,
   .extra2  = &one,
  },
+ {
+  .procname = "sem_next_id",
+  .data  = &init_ipc_ns.ids[IPC_SEM_IDS].next_id,
+  .maxlen  = sizeof(init_ipc_ns.ids[IPC_SEM_IDS].next_id),
+  .mode  = 0644,
+  .proc_handler = proc_ipc_dointvec_minmax,
+  .extra1  = &zero,
+  .extra2  = &int_max,
+ },
+ {
+  .procname = "msg_next_id",
+  .data  = &init_ipc_ns.ids[IPC_MSG_IDS].next_id,
+  .maxlen  = sizeof(init_ipc_ns.ids[IPC_MSG_IDS].next_id),
+  .mode  = 0644,
+  .proc_handler = proc_ipc_dointvec_minmax,
+  .extra1  = &zero,
+  .extra2  = &int_max,
+ },
+ {
+  .procname = "shm_next_id",
+  .data  = &init_ipc_ns.ids[IPC_SHM_IDS].next_id,
+  .maxlen  = sizeof(init_ipc_ns.ids[IPC_SHM_IDS].next_id),
+  .mode  = 0644,
+  .proc_handler = proc_ipc_dointvec_minmax,
+  .extra1  = &zero,
+  .extra2  = &int_max,
+ },
  {}
 };

diff --git a/ipc/util.c b/ipc/util.c
index 72fd078..a961e46 100644
--- a/ipc/util.c
+++ b/ipc/util.c
@@ -122,6 +122,7 @@ void ipc_init_ids(struct ipc_ids *ids)

  ids->in_use = 0;
  ids->seq = 0;
+ ids->next_id = -1;
  {
   int seq_limit = INT_MAX/SEQ_MULTIPLIER;
   if (seq_limit > USHRT_MAX)
@@ -252,6 +253,7 @@ int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size)
```

```
  kuid_t euid;
  kgid_t egid;
  int id, err;
+ int next_id = ids->next_id;

  if (size > IPCMNI)
   size = IPCMNI;
@@ -264,7 +266,8 @@ 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,
+     (next_id < 0) ? 0 : ipcid_to_idx(next_id), &id);
  if (err) {
   spin_unlock(&new->lock);
   rcu_read_unlock();
@@ -277,9 +280,14 @@ int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size)
  new->cuid = new->uid = euid;
  new->gid = new->cgid = egid;

- new->seq = ids->seq++;
- if(ids->seq > ids->seq_max)
-  ids->seq = 0;
+ if (next_id < 0) {
+  new->seq = ids->seq++;
+  if(ids->seq > ids->seq_max)
+   ids->seq = 0;
+ } else {
+  new->seq = ipcid_to_seqx(next_id);
+  ids->next_id = -1;
+ }

  new->id = ipc_buildid(id, new->seq);
  return id;
diff --git a/ipc/util.h b/ipc/util.h
index c8fe2f7..a61e0ca 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_seqx(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);
```