Subject: [PATCH 2/6] IPC namespace - utils
Posted by Kirill Korotaev on Fri, 09 Jun 2006 15:05:29 GMT
View Forum Message <> Reply to Message

This patch adds basic IPC namespace functionality to
IPC utils:
- init_ipc_ns
- copy/clone/unshare/free IPC ns
- /proc preparations

Signed-Off-By: Pavel Emelianov <xemul@openvz.org>
Signed-Off-By: Kirill Korotaev <dev@openvz.org>

```
--- ./ipc/util.c.ipcns 2006-05-18 12:05:15.000000000 +0400
+++ ./ipc/util.c 2006-06-09 14:19:30.000000000 +0400
@@ -12,6 +12,9 @@
  *            Mingming Cao <cmm@us.ibm.com>
  * Mar 2006 - support for audit of ipc object properties
  *            Dustin Kirkland <dustin.kirkland@us.ibm.com>
+ * Jun 2006 - namespaces ssupport
+ *            OpenVZ, SWsoft Inc.
+ *            Pavel Emelianov <xemul@openvz.org>
  */

 #include <linux/config.h>
@@ -30,6 +33,7 @@
 #include <linux/seq_file.h>
 #include <linux/proc_fs.h>
 #include <linux/audit.h>
+#include <linux/nsproxy.h>

 #include <asm/unistd.h>

@@ -38,10 +42,126 @@
 struct ipc_proc_iface {
   const char *path;
   const char *header;
- struct ipc_ids *ids;
+ int ids;
   int (*show)(struct seq_file *, void *);
 };

+struct ipc_namespace init_ipc_ns = {
+ .kref = {
+   .refcount = ATOMIC_INIT(2),
+ },
+};
+
```

```
+#ifdef CONFIG_IPC_NS
+static struct ipc_namespace *clone_ipc_ns(struct ipc_namespace *old_ns)
+{
+ int err;
+ struct ipc_namespace *ns;
+
+ err = -ENOMEM;
+ ns = kmalloc(sizeof(struct ipc_namespace), GFP_KERNEL);
+ if (ns == NULL)
+  goto err_mem;
+
+ err = sem_init_ns(ns);
+ if (err)
+  goto err_sem;
+ err = msg_init_ns(ns);
+ if (err)
+  goto err_msg;
+ err = shm_init_ns(ns);
+ if (err)
+  goto err_shm;
+
+ kref_init(&ns->kref);
+ return ns;
+
+err_shm:
+ msg_exit_ns(ns);
+err_msg:
+ sem_exit_ns(ns);
+err_sem:
+ kfree(ns);
+err_mem:
+ return ERR_PTR(err);
+}
+
+int unshare_ipcs(unsigned long unshare_flags, struct ipc_namespace **new_ipc)
+{
+ struct ipc_namespace *new;
+
+ if (unshare_flags & CLONE_NEWIPC) {
+  if (!capable(CAP_SYS_ADMIN))
+   return -EPERM;
+
+  new = clone_ipc_ns(current->nsproxy->ipc_ns);
+  if (IS_ERR(new))
+   return PTR_ERR(new);
+
+  *new_ipc = new;
+ }
```

```
+
+ return 0;
+}
+
+int copy_ipcs(unsigned long flags, struct task_struct *tsk)
+{
+ struct ipc_namespace *old_ns = tsk->nsproxy->ipc_ns;
+ struct ipc_namespace *new_ns;
+ int err = 0;
+
+ if (!old_ns)
+  return 0;
+
+ get_ipc_ns(old_ns);
+
+ if (!(flags & CLONE_NEWIPC))
+  return 0;
+
+ if (!capable(CAP_SYS_ADMIN)) {
+  err = -EPERM;
+  goto out;
+ }
+
+ new_ns = clone_ipc_ns(old_ns);
+ if (!new_ns) {
+  err = -ENOMEM;
+  goto out;
+ }
+
+ tsk->nsproxy->ipc_ns = new_ns;
+out:
+ put_ipc_ns(old_ns);
+ return err;
+}
+
+void free_ipc_ns(struct kref *kref)
+{
+ struct ipc_namespace *ns;
+
+ ns = container_of(kref, struct ipc_namespace, kref);
+ sem_exit_ns(ns);
+ msg_exit_ns(ns);
+ shm_exit_ns(ns);
+ kfree(ns);
+}
+#else
+int unshare_ipcs(unsigned long flags, struct ipc_namespace **ns)
+{
```

```
+ return -EINVAL;
+}
+
+int copy_ipcs(unsigned long flags, struct task_struct *tsk)
+{
+ return 0;
+}
+
+void free_ipc_ns(struct kref *kref)
+{
+ BUG(); /* init_ipc_ns should never be put */
+}
+#endif
+
 /**
  * ipc_init - initialise IPC subsystem
  *
@@ -68,7 +188,7 @@ __initcall(ipc_init);
  * array itself.
  */

-void __init ipc_init_ids(struct ipc_ids* ids, int size)
+void __ipc_init ipc_init_ids(struct ipc_ids* ids, int size)
 {
  int i;

@@ -111,8 +231,7 @@ static struct file_operations sysvipc_pr
  * @show: show routine.
  */
 void __init ipc_init_proc_interface(const char *path, const char *header,
-        struct ipc_ids *ids,
-        int (*show)(struct seq_file *, void *))
+  int ids, int (*show)(struct seq_file *, void *))
 {
  struct proc_dir_entry *pde;
  struct ipc_proc_iface *iface;
@@ -636,6 +755,9 @@ static void *sysvipc_proc_next(struct se
  struct ipc_proc_iface *iface = s->private;
  struct kern_ipc_perm *ipc = it;
  loff_t p;
+ struct ipc_ids *ids;
+
+ ids = current->nsproxy->ipc_ns->ids[iface->ids];

  /* If we had an ipc id locked before, unlock it */
  if (ipc && ipc != SEQ_START_TOKEN)
@@ -645,8 +767,8 @@ static void *sysvipc_proc_next(struct se
   * p = *pos - 1 (because id 0 starts at position 1)
```

```
   *          + 1 (because we increment the position by one)
   */
- for (p = *pos; p <= iface->ids->max_id; p++) {
-  if ((ipc = ipc_lock(iface->ids, p)) != NULL) {
+ for (p = *pos; p <= ids->max_id; p++) {
+  if ((ipc = ipc_lock(ids, p)) != NULL) {
    *pos = p + 1;
    return ipc;
   }
@@ -665,12 +787,15 @@ static void *sysvipc_proc_start(struct s
  struct ipc_proc_iface *iface = s->private;
  struct kern_ipc_perm *ipc;
  loff_t p;
+ struct ipc_ids *ids;
+
+ ids = current->nsproxy->ipc_ns->ids[iface->ids];

  /*
   * Take the lock - this will be released by the corresponding
   * call to stop().
   */
- mutex_lock(&iface->ids->mutex);
+ mutex_lock(&ids->mutex);

  /* pos < 0 is invalid */
  if (*pos < 0)
@@ -681,8 +806,8 @@ static void *sysvipc_proc_start(struct s
   return SEQ_START_TOKEN;

  /* Find the (pos-1)th ipc */
- for (p = *pos - 1; p <= iface->ids->max_id; p++) {
-  if ((ipc = ipc_lock(iface->ids, p)) != NULL) {
+ for (p = *pos - 1; p <= ids->max_id; p++) {
+  if ((ipc = ipc_lock(ids, p)) != NULL) {
    *pos = p + 1;
    return ipc;
   }
@@ -694,13 +819,15 @@ static void sysvipc_proc_stop(struct seq
 {
  struct kern_ipc_perm *ipc = it;
  struct ipc_proc_iface *iface = s->private;
+ struct ipc_ids *ids;

  /* If we had a locked segment, release it */
  if (ipc && ipc != SEQ_START_TOKEN)
   ipc_unlock(ipc);

+ ids = current->nsproxy->ipc_ns->ids[iface->ids];
```

```
  /* Release the lock we took in start() */
- mutex_unlock(&iface->ids->mutex);
+ mutex_unlock(&ids->mutex);
 }

 static int sysvipc_proc_show(struct seq_file *s, void *it)
--- ./ipc/util.h.ipcns 2006-04-21 11:59:36.000000000 +0400
+++ ./ipc/util.h 2006-06-09 14:24:40.000000000 +0400
@@ -3,6 +3,8 @@
  * Copyright (C) 1999 Christoph Rohland
  *
  * ipc helper functions (c) 1999 Manfred Spraul <manfred@colorfullife.com>
+ * namespaces support.      2006 OpenVZ, SWsoft Inc.
+ *                          Pavel Emelianov <xemul@openvz.org>
  */

 #ifndef _IPC_UTIL_H
@@ -15,6 +17,14 @@ void sem_init (void);
 void msg_init (void);
 void shm_init (void);

+int sem_init_ns(struct ipc_namespace *ns);
+int msg_init_ns(struct ipc_namespace *ns);
+int shm_init_ns(struct ipc_namespace *ns);
+
+void sem_exit_ns(struct ipc_namespace *ns);
+void msg_exit_ns(struct ipc_namespace *ns);
+void shm_exit_ns(struct ipc_namespace *ns);
+
 struct ipc_id_ary {
  int size;
  struct kern_ipc_perm *p[0];
@@ -31,15 +41,23 @@ struct ipc_ids {
 };

 struct seq_file;
-void __init ipc_init_ids(struct ipc_ids* ids, int size);
+#ifdef CONFIG_IPC_NS
+#define __ipc_init
+#else
+#define __ipc_init __init
+#endif
+void __ipc_init ipc_init_ids(struct ipc_ids *ids, int size);
 #ifdef CONFIG_PROC_FS
 void __init ipc_init_proc_interface(const char *path, const char *header,
-       struct ipc_ids *ids,
-       int (*show)(struct seq_file *, void *));
+  int ids, int (*show)(struct seq_file *, void *));
```

```
 #else
 #define ipc_init_proc_interface(path, header, ids, show) do {} while (0)
 #endif

+#define IPC_SEM_IDS 0
+#define IPC_MSG_IDS 1
+#define IPC_SHM_IDS 2
+
 /* must be called with ids->mutex acquired.*/
 int ipc_findkey(struct ipc_ids* ids, key_t key);
 int ipc_addid(struct ipc_ids* ids, struct kern_ipc_perm* new, int size);
```