Subject: [PATCH 3/6] IPC namespace - msg
Posted by Kirill Korotaev on Fri, 09 Jun 2006 15:07:05 GMT
View Forum Message <> Reply to Message

IPC namespace support for IPC msg code.

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

```
--- ./ipc/msg.c.ipcns 2006-06-06 14:47:58.000000000 +0400
+++ ./ipc/msg.c 2006-06-09 14:20:57.000000000 +0400
@@ -16,6 +16,10 @@
  *
  * support for audit of ipc object properties and permission changes
  * Dustin Kirkland <dustin.kirkland@us.ibm.com>
+ *
+ * namespaces support
+ * OpenVZ, SWsoft Inc.
+ * Pavel Emelianov <xemul@openvz.org>
  */

 #include <linux/capability.h>
@@ -32,16 +36,12 @@
 #include <linux/audit.h>
 #include <linux/seq_file.h>
 #include <linux/mutex.h>
+#include <linux/nsproxy.h>

 #include <asm/current.h>
 #include <asm/uaccess.h>
 #include "util.h"

-/* sysctl: */
-int msg_ctlmax = MSGMAX;
-int msg_ctlmnb = MSGMNB;
-int msg_ctlmni = MSGMNI;
-
 /* one msg_receiver structure for each sleeping receiver */
 struct msg_receiver {
  struct list_head r_list;
@@ -68,32 +68,75 @@ struct msg_sender {
 static atomic_t msg_bytes = ATOMIC_INIT(0);
 static atomic_t msg_hdrs = ATOMIC_INIT(0);

-static struct ipc_ids msg_ids;
+static struct ipc_ids init_msg_ids;
```

```
-#define msg_lock(id) ((struct msg_queue*)ipc_lock(&msg_ids,id))
-#define msg_unlock(msq) ipc_unlock(&(msq)->q_perm)
-#define msg_rmid(id) ((struct msg_queue*)ipc_rmid(&msg_ids,id))
-#define msg_checkid(msq, msgid) \
- ipc_checkid(&msg_ids,&msq->q_perm,msgid)
-#define msg_buildid(id, seq) \
- ipc_buildid(&msg_ids, id, seq)
+#define msg_ids(ns) (*((ns)->ids[IPC_MSG_IDS]))

-static void freeque (struct msg_queue *msq, int id);
-static int newque (key_t key, int msgflg);
+#define msg_lock(ns, id) ((struct msg_queue*)ipc_lock(&msg_ids(ns), id))
+#define msg_unlock(msq)  ipc_unlock(&(msq)->q_perm)
+#define msg_rmid(ns, id) ((struct msg_queue*)ipc_rmid(&msg_ids(ns),id))
+#define msg_checkid(ns, msq, msgid) \
+ ipc_checkid(&msg_ids(ns),&msq->q_perm,msgid)
+#define msg_buildid(ns, id, seq) \
+ ipc_buildid(&msg_ids(ns), id, seq)
+
+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
 static int sysvipc_msg_proc_show(struct seq_file *s, void *it);
 #endif

+static void __ipc_init __msg_init_ns(struct ipc_namespace *ns, struct ipc_ids *ids)
+{
+ ns->ids[IPC_MSG_IDS] = ids;
+ ns->msg_ctlmax = MSGMAX;
+ ns->msg_ctlmnb = MSGMNB;
+ ns->msg_ctlmni = MSGMNI;
+ ipc_init_ids(ids, ns->msg_ctlmni);
+}
+
+#ifdef CONFIG_IPC_NS
+int msg_init_ns(struct ipc_namespace *ns)
+{
+ struct ipc_ids *ids;
+
+ ids = kmalloc(sizeof(struct ipc_ids), GFP_KERNEL);
+ if (ids == NULL)
+  return -ENOMEM;
+
+ __msg_init_ns(ns, ids);
+ return 0;
+}
+
+void msg_exit_ns(struct ipc_namespace *ns)
```

```
+{
+ int i;
+ struct msg_queue *msq;
+
+ mutex_lock(&msg_ids(ns).mutex);
+ for (i = 0; i <= msg_ids(ns).max_id; i++) {
+  msq = msg_lock(ns, i);
+  if (msq == NULL)
+   continue;
+
+  freeque(ns, msq, i);
+ }
+ mutex_unlock(&msg_ids(ns).mutex);
+
+ kfree(ns->ids[IPC_MSG_IDS]);
+ ns->ids[IPC_MSG_IDS] = NULL;
+}
+#endif
+
 void __init msg_init (void)
 {
- ipc_init_ids(&msg_ids,msg_ctlmni);
+ __msg_init_ns(&init_ipc_ns, &init_msg_ids);
  ipc_init_proc_interface("sysvipc/msg",
     "       key      msqid perms      cbytes       qnum lspid lrpid   uid   gid  cuid  cgid      stime      rtime
     ctime\n",
-    &msg_ids,
-    sysvipc_msg_proc_show);
+    IPC_MSG_IDS, sysvipc_msg_proc_show);
 }

-static int newque (key_t key, int msgflg)
+static int newque (struct ipc_namespace *ns, key_t key, int msgflg)
 {
  int id;
  int retval;
@@ -113,18 +156,18 @@ static int newque (key_t key, int msgflg
   return retval;
  }

- id = ipc_addid(&msg_ids, &msq->q_perm, msg_ctlmni);
+ id = ipc_addid(&msg_ids(ns), &msq->q_perm, ns->msg_ctlmni);
  if(id == -1) {
   security_msg_queue_free(msq);
   ipc_rcu_putref(msq);
   return -ENOSPC;
  }
```

```
- msq->q_id = msg_buildid(id,msq->q_perm.seq);
+ msq->q_id = msg_buildid(ns, id, msq->q_perm.seq);
  msq->q_stime = msq->q_rtime = 0;
  msq->q_ctime = get_seconds();
  msq->q_cbytes = msq->q_qnum = 0;
- msq->q_qbytes = msg_ctlmnb;
+ msq->q_qbytes = ns->msg_ctlmnb;
  msq->q_lspid = msq->q_lrpid = 0;
  INIT_LIST_HEAD(&msq->q_messages);
  INIT_LIST_HEAD(&msq->q_receivers);
@@ -187,13 +230,13 @@ static void expunge_all(struct msg_queue
  * msg_ids.mutex and the spinlock for this message queue is hold
  * before freeque() is called. msg_ids.mutex remains locked on exit.
  */
-static void freeque (struct msg_queue *msq, int id)
+static void freeque (struct ipc_namespace *ns, struct msg_queue *msq, int id)
 {
  struct list_head *tmp;

  expunge_all(msq,-EIDRM);
  ss_wakeup(&msq->q_senders,1);
- msq = msg_rmid(id);
+ msq = msg_rmid(ns, id);
  msg_unlock(msq);

  tmp = msq->q_messages.next;
@@ -212,31 +255,34 @@ asmlinkage long sys_msgget (key_t key, i
 {
  int id, ret = -EPERM;
  struct msg_queue *msq;
+ struct ipc_namespace *ns;
+
+ ns = current->nsproxy->ipc_ns;

- mutex_lock(&msg_ids.mutex);
+ mutex_lock(&msg_ids(ns).mutex);
  if (key == IPC_PRIVATE)
-  ret = newque(key, msgflg);
- else if ((id = ipc_findkey(&msg_ids, key)) == -1) { /* key not used */
+  ret = newque(ns, key, msgflg);
+ else if ((id = ipc_findkey(&msg_ids(ns), key)) == -1) { /* key not used */
   if (!(msgflg & IPC_CREAT))
    ret = -ENOENT;
   else
-   ret = newque(key, msgflg);
+   ret = newque(ns, key, msgflg);
  } else if (msgflg & IPC_CREAT && msgflg & IPC_EXCL) {
   ret = -EEXIST;
```

```
  } else {
-  msq = msg_lock(id);
+  msq = msg_lock(ns, id);
    BUG_ON(msq==NULL);
    if (ipcperms(&msq->q_perm, msgflg))
     ret = -EACCES;
    else {
-    int qid = msg_buildid(id, msq->q_perm.seq);
+    int qid = msg_buildid(ns, id, msq->q_perm.seq);
         ret = security_msg_queue_associate(msq, msgflg);
     if (!ret)
      ret = qid;
    }
    msg_unlock(msq);
  }
- mutex_unlock(&msg_ids.mutex);
+ mutex_unlock(&msg_ids(ns).mutex);
  return ret;
 }

@@ -337,11 +383,13 @@ asmlinkage long sys_msgctl (int msqid, i
  struct msg_queue *msq;
  struct msq_setbuf setbuf;
  struct kern_ipc_perm *ipcp;
+ struct ipc_namespace *ns;

  if (msqid < 0 || cmd < 0)
   return -EINVAL;

  version = ipc_parse_version(&cmd);
+ ns = current->nsproxy->ipc_ns;

  switch (cmd) {
  case IPC_INFO:
@@ -361,14 +409,14 @@ asmlinkage long sys_msgctl (int msqid, i
    return err;

    memset(&msginfo,0,sizeof(msginfo));
-  msginfo.msgmni = msg_ctlmni;
-  msginfo.msgmax = msg_ctlmax;
-  msginfo.msgmnb = msg_ctlmnb;
+  msginfo.msgmni = ns->msg_ctlmni;
+  msginfo.msgmax = ns->msg_ctlmax;
+  msginfo.msgmnb = ns->msg_ctlmnb;
    msginfo.msgssz = MSGSSZ;
    msginfo.msgseg = MSGSEG;
-  mutex_lock(&msg_ids.mutex);
+  mutex_lock(&msg_ids(ns).mutex);
```

```
   if (cmd == MSG_INFO) {
-   msginfo.msgpool = msg_ids.in_use;
+   msginfo.msgpool = msg_ids(ns).in_use;
    msginfo.msgmap = atomic_read(&msg_hdrs);
    msginfo.msgtql = atomic_read(&msg_bytes);
   } else {
@@ -376,8 +424,8 @@ asmlinkage long sys_msgctl (int msqid, i
    msginfo.msgpool = MSGPOOL;
    msginfo.msgtql = MSGTQL;
   }
-  max_id = msg_ids.max_id;
-  mutex_unlock(&msg_ids.mutex);
+  max_id = msg_ids(ns).max_id;
+  mutex_unlock(&msg_ids(ns).mutex);
   if (copy_to_user (buf, &msginfo, sizeof(struct msginfo)))
    return -EFAULT;
   return (max_id < 0) ? 0: max_id;
@@ -389,20 +437,20 @@ asmlinkage long sys_msgctl (int msqid, i
   int success_return;
   if (!buf)
    return -EFAULT;
-  if(cmd == MSG_STAT && msqid >= msg_ids.entries->size)
+  if(cmd == MSG_STAT && msqid >= msg_ids(ns).entries->size)
    return -EINVAL;

   memset(&tbuf,0,sizeof(tbuf));

-  msq = msg_lock(msqid);
+  msq = msg_lock(ns, msqid);
   if (msq == NULL)
    return -EINVAL;

   if(cmd == MSG_STAT) {
-   success_return = msg_buildid(msqid, msq->q_perm.seq);
+   success_return = msg_buildid(ns, msqid, msq->q_perm.seq);
   } else {
    err = -EIDRM;
-   if (msg_checkid(msq,msqid))
+   if (msg_checkid(ns, msq,msqid))
     goto out_unlock;
    success_return = 0;
   }
@@ -440,14 +488,14 @@ asmlinkage long sys_msgctl (int msqid, i
   return  -EINVAL;
  }

- mutex_lock(&msg_ids.mutex);
- msq = msg_lock(msqid);
```

```
+ mutex_lock(&msg_ids(ns).mutex);
+ msq = msg_lock(ns, msqid);
  err=-EINVAL;
  if (msq == NULL)
   goto out_up;

  err = -EIDRM;
- if (msg_checkid(msq,msqid))
+ if (msg_checkid(ns, msq,msqid))
   goto out_unlock_up;
  ipcp = &msq->q_perm;

@@ -474,7 +522,7 @@ asmlinkage long sys_msgctl (int msqid, i
  case IPC_SET:
  {
   err = -EPERM;
-  if (setbuf.qbytes > msg_ctlmnb && !capable(CAP_SYS_RESOURCE))
+  if (setbuf.qbytes > ns->msg_ctlmnb && !capable(CAP_SYS_RESOURCE))
    goto out_unlock_up;

   msq->q_qbytes = setbuf.qbytes;
@@ -496,12 +544,12 @@ asmlinkage long sys_msgctl (int msqid, i
   break;
  }
  case IPC_RMID:
-  freeque (msq, msqid);
+  freeque (ns, msq, msqid);
   break;
  }
  err = 0;
 out_up:
- mutex_unlock(&msg_ids.mutex);
+ mutex_unlock(&msg_ids(ns).mutex);
  return err;
 out_unlock_up:
  msg_unlock(msq);
@@ -570,8 +618,11 @@ asmlinkage long sys_msgsnd (int msqid, s
  struct msg_msg *msg;
  long mtype;
  int err;
+ struct ipc_namespace *ns;
+
+ ns = current->nsproxy->ipc_ns;

- if (msgsz > msg_ctlmax || (long) msgsz < 0 || msqid < 0)
+ if (msgsz > ns->msg_ctlmax || (long) msgsz < 0 || msqid < 0)
   return -EINVAL;
  if (get_user(mtype, &msgp->mtype))
```

```
   return -EFAULT;
@@ -585,13 +636,13 @@ asmlinkage long sys_msgsnd (int msqid, s
  msg->m_type = mtype;
  msg->m_ts = msgsz;

- msq = msg_lock(msqid);
+ msq = msg_lock(ns, msqid);
  err=-EINVAL;
  if(msq==NULL)
   goto out_free;

  err= -EIDRM;
- if (msg_checkid(msq,msqid))
+ if (msg_checkid(ns, msq,msqid))
   goto out_unlock_free;

  for (;;) {
@@ -682,17 +733,19 @@ asmlinkage long sys_msgrcv (int msqid, s
  struct msg_queue *msq;
  struct msg_msg *msg;
  int mode;
+ struct ipc_namespace *ns;

  if (msqid < 0 || (long) msgsz < 0)
   return -EINVAL;
  mode = convert_mode(&msgtyp,msgflg);
+ ns = current->nsproxy->ipc_ns;

- msq = msg_lock(msqid);
+ msq = msg_lock(ns, msqid);
  if(msq==NULL)
   return -EINVAL;

  msg = ERR_PTR(-EIDRM);
- if (msg_checkid(msq,msqid))
+ if (msg_checkid(ns, msq, msqid))
   goto out_unlock;

  for (;;) {
```