
Subject: [PATCH 2.6.24-rc8-mm1 03/15] IPC/message queues: introduce msgctl_down

Posted by [Pierre Peiffer](#) on Tue, 29 Jan 2008 16:02:32 GMT

[View Forum Message](#) <> [Reply to Message](#)

From: Pierre Peiffer <pierre.peiffer@bull.net>

Currently, sys_msgctl is not easy to read.

This patch tries to improve that by introducing the msgctl_down function to handle all commands requiring the rwmutex to be taken in write mode (ie IPC_SET and IPC_RMID for now). It is the equivalent function of semctl_down for message queues.

This greatly changes the readability of sys_msgctl and also harmonizes the way these commands are handled among all IPCs.

Signed-off-by: Pierre Peiffer <pierre.peiffer@bull.net>

Acked-by: Serge Hallyn <serue@us.ibm.com>

ipc/msg.c | 162 ++++++-----
1 file changed, 89 insertions(+), 73 deletions(-)

Index: b/ipc/msg.c

=====

--- a/ipc/msg.c

+++ b/ipc/msg.c

@@ -399,10 +399,95 @@ copy_msqid_from_user(struct msq_setbuf *
 }
 }

-asmlinkage long sys_msgctl(int msqid, int cmd, struct msqid_ds __user *buf)

+/*

+ * This function handles some msgctl commands which require the rw_mutex

+ * to be held in write mode.

+ * NOTE: no locks must be held, the rw_mutex is taken inside this function.

+ */

+static int msgctl_down(struct ipc_namespace *ns, int msqid, int cmd,

+ struct msqid_ds __user *buf, int version)

{

struct kern_ipc_perm *ipcp;

- struct msq_setbuf uninitialized_var(setbuf);

+ struct msq_setbuf setbuf;

+ struct msg_queue *msq;

+ int err;

+

+ if (cmd == IPC_SET) {

```

+ if (copy_msqid_from_user(&setbuf, buf, version))
+ return -EFAULT;
+ }
+
+ down_write(&msg_ids(ns).rw_mutex);
+ msq = msg_lock_check_down(ns, msqid);
+ if (IS_ERR(msq)) {
+ err = PTR_ERR(msq);
+ goto out_up;
+ }
+
+ ipcp = &msq->q_perm;
+
+ err = audit_ipc_obj(ipcp);
+ if (err)
+ goto out_unlock;
+
+ if (cmd == IPC_SET) {
+ err = audit_ipc_set_perm(setbuf.qbytes, setbuf.uid, setbuf.gid,
+ setbuf.mode);
+ if (err)
+ goto out_unlock;
+ }
+
+ if (current->euid != ipcp->cuid &&
+ current->euid != ipcp->uid &&
+ !capable(CAP_SYS_ADMIN)) {
+ /* We _could_ check for CAP_CHOWN above, but we don't */
+ err = -EPERM;
+ goto out_unlock;
+ }
+
+ err = security_msg_queue_msgctl(msq, cmd);
+ if (err)
+ goto out_unlock;
+
+ switch (cmd) {
+ case IPC_RMID:
+ freeque(ns, ipcp);
+ goto out_up;
+ case IPC_SET:
+ if (setbuf.qbytes > ns->msg_ctlmnb &&
+ !capable(CAP_SYS_RESOURCE)) {
+ err = -EPERM;
+ goto out_unlock;
+ }
+
+ msq->q_qbytes = setbuf.qbytes;

```

```

+
+ ipc->uid = setbuf.uid;
+ ipc->gid = setbuf.gid;
+ ipc->mode = (ipc->mode & ~S_IRWXUGO) |
+   (S_IRWXUGO & setbuf.mode);
+ msq->q_ctime = get_seconds();
+ /* sleeping receivers might be excluded by
+  * stricter permissions.
+  */
+ expunge_all(msq, -EAGAIN);
+ /* sleeping senders might be able to send
+  * due to a larger queue size.
+  */
+ ss_wakeup(&msq->q_senders, 0);
+ break;
+ default:
+   err = -EINVAL;
+ }
+out_unlock:
+ msg_unlock(msq);
+out_up:
+ up_write(&msg_ids(ns).rw_mutex);
+ return err;
+}
+
+asmlinkage long sys_msgctl(int msqid, int cmd, struct msqid_ds __user *buf)
+{
+   struct msg_queue *msq;
+   int err, version;
+   struct ipc_namespace *ns;
@@ -498,82 +583,13 @@ asmlinkage long sys_msgctl(int msqid, in
+   return success_return;
+ }
+   case IPC_SET:
+   - if (!buf)
+   -   return -EFAULT;
+   - if (copy_msqid_from_user(&setbuf, buf, version))
+   -   return -EFAULT;
+   - break;
+   case IPC_RMID:
+   - break;
+   err = msgctl_down(ns, msqid, cmd, buf, version);
+   return err;
+   default:
+   return -EINVAL;
+ }
+
+   down_write(&msg_ids(ns).rw_mutex);

```

```

- msq = msg_lock_check_down(ns, msqid);
- if (IS_ERR(msq)) {
-   err = PTR_ERR(msq);
-   goto out_up;
- }
-
- ipcp = &msq->q_perm;
-
- err = audit_ipc_obj(ipcp);
- if (err)
-   goto out_unlock_up;
- if (cmd == IPC_SET) {
-   err = audit_ipc_set_perm(setbuf.qbytes, setbuf.uid, setbuf.gid,
-     setbuf.mode);
-   if (err)
-     goto out_unlock_up;
- }
-
- err = -EPERM;
- if (current->euid != ipcp->cuid &&
-     current->euid != ipcp->uid && !capable(CAP_SYS_ADMIN))
-   /* We _could_ check for CAP_CHOWN above, but we don't */
-   goto out_unlock_up;
-
- err = security_msg_queue_msgctl(msq, cmd);
- if (err)
-   goto out_unlock_up;
-
- switch (cmd) {
- case IPC_SET:
- {
-   err = -EPERM;
-   if (setbuf.qbytes > ns->msg_ctlmnb && !capable(CAP_SYS_RESOURCE))
-     goto out_unlock_up;
- }
-
-   msq->q_qbytes = setbuf.qbytes;
-
-   ipcp->uid = setbuf.uid;
-   ipcp->gid = setbuf.gid;
-   ipcp->mode = (ipcp->mode & ~S_IRWXUGO) |
-     (S_IRWXUGO & setbuf.mode);
-   msq->q_ctime = get_seconds();
-   /* sleeping receivers might be excluded by
-    * stricter permissions.
-    */
-   expunge_all(msq, -EAGAIN);
-   /* sleeping senders might be able to send
-    * due to a larger queue size.

```

```
- */
- ss_wakeup(&msq->q_senders, 0);
- msg_unlock(msq);
- break;
- }
- case IPC_RMID:
- freeque(ns, &msq->q_perm);
- break;
- }
- err = 0;
-out_up:
- up_write(&msg_ids(ns).rw_mutex);
- return err;
-out_unlock_up:
- msg_unlock(msq);
- goto out_up;
out_unlock:
- msg_unlock(msq);
- return err;
```

--

Pierre Peiffer

Containers mailing list
Containers@lists.linux-foundation.org
<https://lists.linux-foundation.org/mailman/listinfo/containers>
