Subject: [patch -mm] update mq_notify to use a struct pid
Posted by Cedric Le Goater on Fri, 08 Sep 2006 16:39:31 GMT
View Forum Message <> Reply to Message

message queues can signal a process waiting for a message.

this patch replaces the pid_t value with a struct pid to avoid pid wrap
around problems.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Cc: Eric Biederman <ebiederm@xmission.com>
Cc: Andrew Morton <akpm@osdl.org>
Cc: containers@lists.osdl.org

---
 ipc/mqueue.c | 27 +++++++++++++++++++------------
 1 file changed, 15 insertions(+), 12 deletions(-)

Index: 2.6.18-rc6-mm1/ipc/mqueue.c
===
--- 2.6.18-rc6-mm1.orig/ipc/mqueue.c
+++ 2.6.18-rc6-mm1/ipc/mqueue.c
@@ -73,7 +73,7 @@ struct mqueue_inode_info {
  struct mq_attr attr;

  struct sigevent notify;
- pid_t notify_owner;
+ struct pid* notify_owner;
  struct user_struct *user; /* user who created, for accounting */
  struct sock *notify_sock;
  struct sk_buff *notify_cookie;
@@ -134,7 +134,7 @@ static struct inode *mqueue_get_inode(st
    INIT_LIST_HEAD(&info->e_wait_q[0].list);
    INIT_LIST_HEAD(&info->e_wait_q[1].list);
    info->messages = NULL;
-   info->notify_owner = 0;
+   info->notify_owner = NULL;
    info->qsize = 0;
    info->user = NULL; /* set when all is ok */
    memset(&info->attr, 0, sizeof(info->attr));
@@ -338,7 +338,7 @@ static ssize_t mqueue_read_file(struct f
    (info->notify_owner &&
     info->notify.sigev_notify == SIGEV_SIGNAL) ?
     info->notify.sigev_signo : 0,
-   info->notify_owner);
+   pid_nr(info->notify_owner));
  spin_unlock(&info->lock);
  buffer[sizeof(buffer)-1] = '\0';

```
  slen = strlen(buffer)+1;
@@ -363,7 +363,7 @@ static int mqueue_flush_file(struct file
  struct mqueue_inode_info *info = MQUEUE_I(filp->f_dentry->d_inode);

  spin_lock(&info->lock);
- if (current->tgid == info->notify_owner)
+ if (task_tgid(current) == info->notify_owner)
   remove_notification(info);

  spin_unlock(&info->lock);
@@ -518,8 +518,8 @@ static void __do_notify(struct mqueue_in
    sig_i.si_pid = current->tgid;
    sig_i.si_uid = current->uid;

-   kill_proc_info(info->notify.sigev_signo,
-          &sig_i, info->notify_owner);
+   kill_pid_info(info->notify.sigev_signo,
+          &sig_i, info->notify_owner);
    break;
   case SIGEV_THREAD:
    set_cookie(info->notify_cookie, NOTIFY_WOKENUP);
@@ -528,7 +528,8 @@ static void __do_notify(struct mqueue_in
    break;
   }
   /* after notification unregisters process */
-  info->notify_owner = 0;
+  put_pid(info->notify_owner);
+  info->notify_owner = NULL;
  }
  wake_up(&info->wait_q);
 }
@@ -566,12 +567,13 @@ static long prepare_timeout(const struct

 static void remove_notification(struct mqueue_inode_info *info)
 {
- if (info->notify_owner != 0 &&
+ if (info->notify_owner != NULL &&
      info->notify.sigev_notify == SIGEV_THREAD) {
   set_cookie(info->notify_cookie, NOTIFY_REMOVED);
   netlink_sendskb(info->notify_sock, info->notify_cookie, 0);
  }
- info->notify_owner = 0;
+ put_pid(info->notify_owner);
+ info->notify_owner = NULL;
 }

 static int mq_attr_ok(struct mq_attr *attr)
@@ -1062,11 +1064,11 @@ retry:
```

```
  ret = 0;
  spin_lock(&info->lock);
  if (u_notification == NULL) {
-  if (info->notify_owner == current->tgid) {
+  if (info->notify_owner == task_tgid(current)) {
     remove_notification(info);
     inode->i_atime = inode->i_ctime = CURRENT_TIME;
    }
- } else if (info->notify_owner != 0) {
+ } else if (info->notify_owner != NULL) {
    ret = -EBUSY;
  } else {
    switch (notification.sigev_notify) {
@@ -1086,7 +1088,8 @@ retry:
     info->notify.sigev_notify = SIGEV_SIGNAL;
     break;
    }
-  info->notify_owner = current->tgid;
+
+  info->notify_owner = get_pid(task_tgid(current));
    inode->i_atime = inode->i_ctime = CURRENT_TIME;
  }
  spin_unlock(&info->lock);
```