Subject: [PATCH v3 07/10] IPC: message queue receive cleanup
Posted by Stanislav Kinsbursky on Fri, 10 Aug 2012 14:25:56 GMT
View Forum Message <> Reply to Message

This patch moves all message related manipulation into one function msg_fill().
Actually, two functions because of the compat one.

Signed-off-by: Stanislav Kinsbursky <skinsbursky@parallels.com>
Signed-off-by: Cyrill Gorcunov <gorcunov@openvz.org>

Conflicts:

```
 arch/tile/kernel/compat.c
 include/linux/compat.h
---
 arch/tile/kernel/compat.c |   29 ++++++++++++++++++++++++++++
 include/linux/compat.h    |    2 ++
 include/linux/msg.h       |    5 +++--
 ipc/compat.c              |   33 +++++++++++++++++++++----------
 ipc/msg.c                 |   44 ++++++++++++++++++++++++++++++--------------
 5 files changed, 74 insertions(+), 39 deletions(-)

diff --git a/arch/tile/kernel/compat.c b/arch/tile/kernel/compat.c
index d67459b..e74d61b 100644
--- a/arch/tile/kernel/compat.c
+++ b/arch/tile/kernel/compat.c
@@ -94,6 +94,35 @@ long compat_sys_sched_rr_get_interval(compat_pid_t pid,
  return ret;
 }

+/*
+ * The usual compat_sys_msgsnd() and _msgrcv() seem to be assuming
+ * some different calling convention than our normal 32-bit tile code.
+ */
+
+/* Already defined in ipc/compat.c, but we need it here. */
+struct compat_msgbuf {
+ compat_long_t mtype;
+ char mtext[1];
+};
+
+long tile_compat_sys_msgsnd(int msqid,
+       struct compat_msgbuf __user *msgp,
+       size_t msgsz, int msgflg)
+{
+ compat_long_t mtype;
+
+ if (get_user(mtype, &msgp->mtype))
```

```
+  return -EFAULT;
+ return do_msgsnd(msqid, mtype, msgp->mtext, msgsz, msgflg);
+}
+
+long tile_compat_sys_msgrcv(int msqid,
+       struct compat_msgbuf __user *msgp,
+       size_t msgsz, long msgtyp, int msgflg)
+{
+ return do_msgrcv(msqid, msgp, msgsz, msgtyp, msgflg, compat_do_msg_fill);
+}
+
 /* Provide the compat syscall number to call mapping. */
 #undef __SYSCALL
 #define __SYSCALL(nr, call) [nr] = (call),
diff --git a/include/linux/compat.h b/include/linux/compat.h
index 4e89039..6e035dc 100644
--- a/include/linux/compat.h
+++ b/include/linux/compat.h
@@ -245,6 +245,7 @@ struct compat_sysctl_args;
 struct compat_kexec_segment;
 struct compat_mq_attr;
 struct compat_msgbuf;
+struct msg_msg;

 extern void compat_exit_robust_list(struct task_struct *curr);

@@ -258,6 +259,7 @@ compat_sys_get_robust_list(int pid, compat_uptr_t __user *head_ptr,
 #ifdef CONFIG_ARCH_WANT_OLD_COMPAT_IPC
 long compat_sys_semctl(int first, int second, int third, void __user *uptr);
 long compat_sys_msgsnd(int first, int second, int third, void __user *uptr);
+long compat_do_msg_fill(void __user *dest, struct msg_msg *msg, size_t bufsz);
 long compat_sys_msgrcv(int first, int second, int msgtyp, int third,
   int version, void __user *uptr);
 long compat_sys_shmat(int first, int second, compat_uptr_t third, int version,
diff --git a/include/linux/msg.h b/include/linux/msg.h
index 6689e73..9411b76 100644
--- a/include/linux/msg.h
+++ b/include/linux/msg.h
@@ -105,8 +105,9 @@ struct msg_queue {
 /* Helper routines for sys_msgsnd and sys_msgrcv */
 extern long do_msgsnd(int msqid, long mtype, void __user *mtext,
    size_t msgsz, int msgflg);
-extern long do_msgrcv(int msqid, long *pmtype, void __user *mtext,
-   size_t msgsz, long msgtyp, int msgflg);
+extern long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
+       int msgflg,
+       long (*msg_fill)(void __user *, struct msg_msg *, size_t ));
```

```
 #endif /* __KERNEL__ */

diff --git a/ipc/compat.c b/ipc/compat.c
index 7cfdb5b..6b07f5c 100644
--- a/ipc/compat.c
+++ b/ipc/compat.c
@@ -341,13 +341,23 @@ long compat_sys_msgsnd(int first, int second, int third, void __user
*uptr)
  return do_msgsnd(first, type, up->mtext, second, third);
 }

+long compat_do_msg_fill(void __user *dest, struct msg_msg *msg, size_t bufsz)
+{
+ struct compat_msgbuf __user *msgp;
+ size_t msgsz;
+
+ if (put_user(msg->m_type, &msgp->mtype))
+  return -EFAULT;
+
+ msgsz = (bufsz > msg->m_ts) ? msg->m_ts : bufsz;
+ if (store_msg(msgp->mtext, msg, msgsz))
+  return -EFAULT;
+ return msgsz;
+}
+
 long compat_sys_msgrcv(int first, int second, int msgtyp, int third,
        int version, void __user *uptr)
 {
- struct compat_msgbuf __user *up;
- long type;
- int err;
-
  if (first < 0)
   return -EINVAL;
  if (second < 0)
@@ -355,23 +365,14 @@ long compat_sys_msgrcv(int first, int second, int msgtyp, int third,

  if (!version) {
   struct compat_ipc_kludge ipck;
-  err = -EINVAL;
   if (!uptr)
-   goto out;
-  err = -EFAULT;
+   return -EINVAL;
   if (copy_from_user (&ipck, uptr, sizeof(ipck)))
-   goto out;
+   return -EFAULT;
   uptr = compat_ptr(ipck.msgp);
```

```
   msgtyp = ipck.msgtyp;
  }
- up = uptr;
- err = do_msgrcv(first, &type, up->mtext, second, msgtyp, third);
- if (err < 0)
-  goto out;
- if (put_user(type, &up->mtype))
-  err = -EFAULT;
-out:
- return err;
+ return do_msgrcv(first, uptr, second, msgtyp, third, compat_do_msg_fill);
 }
 #else
 long compat_sys_semctl(int semid, int semnum, int cmd, int arg)
diff --git a/ipc/msg.c b/ipc/msg.c
index 6bd66d3..08009f5 100644
--- a/ipc/msg.c
+++ b/ipc/msg.c
@@ -762,15 +762,30 @@ static inline int convert_mode(long *msgtyp, int msgflg)
  return SEARCH_EQUAL;
 }

-long do_msgrcv(int msqid, long *pmtype, void __user *mtext,
-  size_t msgsz, long msgtyp, int msgflg)
+static long do_msg_fill(void __user *dest, struct msg_msg *msg, size_t bufsz)
+{
+ struct msgbuf __user *msgp = dest;
+ size_t msgsz;
+
+ if (put_user(msg->m_type, &msgp->mtype))
+  return -EFAULT;
+
+ msgsz = (bufsz > msg->m_ts) ? msg->m_ts : bufsz;
+ if (store_msg(msgp->mtext, msg, msgsz))
+  return -EFAULT;
+ return msgsz;
+}
+
+long do_msgrcv(int msqid, void __user *buf, size_t bufsz, long msgtyp,
+       int msgflg,
+       long (*msg_handler)(void __user *, struct msg_msg *, size_t ))
 {
  struct msg_queue *msq;
  struct msg_msg *msg;
  int mode;
  struct ipc_namespace *ns;

- if (msqid < 0 || (long) msgsz < 0)
```

```
+ if (msqid < 0 || (long) bufsz < 0)
   return -EINVAL;
  mode = convert_mode(&msgtyp, msgflg);
  ns = current->nsproxy->ipc_ns;
@@ -813,7 +828,7 @@ long do_msgrcv(int msqid, long *pmtype, void __user *mtext,
     * Found a suitable message.
     * Unlink it from the queue.
     */
-   if ((msgsz < msg->m_ts) && !(msgflg & MSG_NOERROR)) {
+   if ((bufsz < msg->m_ts) && !(msgflg & MSG_NOERROR)) {
     msg = ERR_PTR(-E2BIG);
     goto out_unlock;
    }
@@ -840,7 +855,7 @@ long do_msgrcv(int msqid, long *pmtype, void __user *mtext,
   if (msgflg & MSG_NOERROR)
    msr_d.r_maxsize = INT_MAX;
   else
-   msr_d.r_maxsize = msgsz;
+   msr_d.r_maxsize = bufsz;
   msr_d.r_msg = ERR_PTR(-EAGAIN);
   current->state = TASK_INTERRUPTIBLE;
   msg_unlock(msq);
@@ -903,29 +918,16 @@ out_unlock:
  if (IS_ERR(msg))
   return PTR_ERR(msg);

- msgsz = (msgsz > msg->m_ts) ? msg->m_ts : msgsz;
- *pmtype = msg->m_type;
- if (store_msg(mtext, msg, msgsz))
-  msgsz = -EFAULT;
-
+ bufsz = msg_handler(buf, msg, bufsz);
  free_msg(msg);

- return msgsz;
+ return bufsz;
 }

 SYSCALL_DEFINE5(msgrcv, int, msqid, struct msgbuf __user *, msgp, size_t, msgsz,
   long, msgtyp, int, msgflg)
 {
- long err, mtype;
-
- err =  do_msgrcv(msqid, &mtype, msgp->mtext, msgsz, msgtyp, msgflg);
- if (err < 0)
-  goto out;
-
- if (put_user(mtype, &msgp->mtype))
```

```
-  err = -EFAULT;
-out:
- return err;
+ return do_msgrcv(msqid, msgp, msgsz, msgtyp, msgflg, do_msg_fill);
 }

 #ifdef CONFIG_PROC_FS
```