Subject: [patch -mm 2/5] mqueue namespace : add unshare support
Posted by Cedric Le Goater on Tue, 02 Oct 2007 08:46:10 GMT
View Forum Message <> Reply to Message

From: Cedric Le Goater <clg@fr.ibm.com>

This patch includes the mqueue namespace in the nsproxy object. It
also adds the support of unshare() and clone() with a new clone flag
CLONE_NEWMQ (1 bit left in the clone flags !)

CLONE_NEWMQ is required to be cloned or unshared along with CLONE_NEWNS.
This is to make sure that no user mounts of the internal mqueue fs
are left behind when the last task exits.

It's totally harmless for the moment because the current code still
uses the default mqueue namespace object 'init_mq_ns'

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
---
 include/linux/init_task.h | 2 ++
 include/linux/nsproxy.h | 2 ++
 include/linux/sched.h | 1 +
 ipc/mqueue.c | 29 +++++++++++++++++++++++++++++-
 kernel/fork.c | 16 ++++++++++++++-
 kernel/nsproxy.c | 15 +++++++++++++--
 6 files changed, 61 insertions(+), 4 deletions(-)

Index: 2.6.23-rc8-mm2/include/linux/init_task.h
===
--- 2.6.23-rc8-mm2.orig/include/linux/init_task.h
+++ 2.6.23-rc8-mm2/include/linux/init_task.h
@@ -10,6 +10,7 @@
 #include <linux/pid_namespace.h>
 #include <linux/user_namespace.h>
 #include <net/net_namespace.h>
+#include <linux/mq_namespace.h>

 #define INIT_FDTABLE \
 {   \
@@ -78,6 +79,7 @@ extern struct nsproxy init_nsproxy;
  INIT_NET_NS(net_ns)   \
  INIT_IPC_NS(ipc_ns)   \
  .user_ns = &init_user_ns,   \
+ INIT_MQ_NS(mq_ns)   \
 }

 #define INIT_SIGHAND(sighand) {   \
Index: 2.6.23-rc8-mm2/include/linux/sched.h

```
===================================================================
--- 2.6.23-rc8-mm2.orig/include/linux/sched.h
+++ 2.6.23-rc8-mm2/include/linux/sched.h
@@ -26,6 +26,7 @@
 #define CLONE_NEWIPC  0x08000000 /* New ipcs */
 #define CLONE_NEWUSER  0x10000000 /* New user namespace */
 #define CLONE_NEWPID  0x20000000 /* New pid namespace */
+#define CLONE_NEWMQ  0x40000000 /* New posix mqueue namespace */

 /*
  * Scheduling policies
Index: 2.6.23-rc8-mm2/kernel/nsproxy.c
===================================================================
--- 2.6.23-rc8-mm2.orig/kernel/nsproxy.c
+++ 2.6.23-rc8-mm2/kernel/nsproxy.c
@@ -85,8 +85,17 @@ static struct nsproxy *create_new_namesp
   goto out_user;
  }

+ new_nsp->mq_ns = copy_mq_ns(flags, tsk->nsproxy->mq_ns);
+ if (IS_ERR(new_nsp->mq_ns)) {
+  err = PTR_ERR(new_nsp->mq_ns);
+  goto out_mq;
+ }
+
  return new_nsp;

+out_mq:
+ if (new_nsp->user_ns)
+  put_user_ns(new_nsp->user_ns);
 out_user:
  if (new_nsp->pid_ns)
   put_pid_ns(new_nsp->pid_ns);
@@ -120,7 +129,7 @@ int copy_namespaces(unsigned long flags,
  get_nsproxy(old_ns);

  if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-     CLONE_NEWUSER | CLONE_NEWPID)))
+         CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWMQ)))
   return 0;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -159,6 +168,8 @@ void free_nsproxy(struct nsproxy *ns)
   put_pid_ns(ns->pid_ns);
  if (ns->user_ns)
   put_user_ns(ns->user_ns);
+ if (ns->mq_ns)
+  put_mq_ns(ns->mq_ns);
```

```
  kmem_cache_free(nsproxy_cachep, ns);
 }

@@ -172,7 +183,7 @@ int unshare_nsproxy_namespaces(unsigned
  int err = 0;

  if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
-          CLONE_NEWUSER)))
+          CLONE_NEWUSER | CLONE_NEWMQ)))
    return 0;

  if (!capable(CAP_SYS_ADMIN))
Index: 2.6.23-rc8-mm2/kernel/fork.c
===================================================================
--- 2.6.23-rc8-mm2.orig/kernel/fork.c
+++ 2.6.23-rc8-mm2/kernel/fork.c
@@ -1002,6 +1002,13 @@ static struct task_struct *copy_process(
  if ((clone_flags & CLONE_SIGHAND) && !(clone_flags & CLONE_VM))
    return ERR_PTR(-EINVAL);

+ /*
+  * mount namespace cannot be unshared when the mqueue
+  * namespace is not
+  */
+ if ((clone_flags & CLONE_NEWMQ) && !(clone_flags & CLONE_NEWNS))
+  return ERR_PTR(-EINVAL);
+
  retval = security_task_create(clone_flags);
  if (retval)
    goto fork_out;
@@ -1552,6 +1559,12 @@ static inline void check_unshare_flags(u
    *flags_ptr |= CLONE_THREAD;

  /*
+  * If unsharing mqueue namespace, must also unshare mnt namespace.
+  */
+ if (*flags_ptr & CLONE_NEWMQ)
+  *flags_ptr |= CLONE_NEWNS;
+
+ /*
    * If unsharing namespace, must also unshare filesystem information.
    */
  if (*flags_ptr & CLONE_NEWNS)
@@ -1668,7 +1681,8 @@ asmlinkage long sys_unshare(unsigned lon
  err = -EINVAL;
  if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
      CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
-    CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER))
```

```
+    CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
+    CLONE_NEWMQ))
   goto bad_unshare_out;

  if ((err = unshare_thread(unshare_flags)))
Index: 2.6.23-rc8-mm2/include/linux/nsproxy.h
===================================================================
--- 2.6.23-rc8-mm2.orig/include/linux/nsproxy.h
+++ 2.6.23-rc8-mm2/include/linux/nsproxy.h
@@ -8,6 +8,7 @@ struct mnt_namespace;
 struct uts_namespace;
 struct ipc_namespace;
 struct pid_namespace;
+struct mq_namespace;

 /*
  * A structure to contain pointers to all per-process
@@ -29,6 +30,7 @@ struct nsproxy {
  struct pid_namespace *pid_ns;
  struct user_namespace *user_ns;
  struct net       *net_ns;
+ struct mq_namespace *mq_ns;
 };
 extern struct nsproxy init_nsproxy;

Index: 2.6.23-rc8-mm2/ipc/mqueue.c
===================================================================
--- 2.6.23-rc8-mm2.orig/ipc/mqueue.c
+++ 2.6.23-rc8-mm2/ipc/mqueue.c
@@ -108,11 +108,38 @@ static inline struct mqueue_inode_info *
  return container_of(inode, struct mqueue_inode_info, vfs_inode);
 }

+static struct mq_namespace *clone_mq_ns(struct mq_namespace *old_ns)
+{
+ struct mq_namespace *mq_ns;
+
+ mq_ns = kmalloc(sizeof(struct mq_namespace), GFP_KERNEL);
+ if (!mq_ns)
+  return ERR_PTR(-ENOMEM);
+
+ kref_init(&mq_ns->kref);
+ mq_ns->queues_count = 0;
+ mq_ns->queues_max = DFLT_QUEUESMAX;
+ mq_ns->msg_max  = DFLT_MSGMAX;
+ mq_ns->msgsize_max = DFLT_MSGSIZEMAX;
+ mq_ns->mnt  = NULL;
+ return mq_ns;
```

```
+}
+
 struct mq_namespace *copy_mq_ns(unsigned long flags,
     struct mq_namespace *old_ns)
 {
+ struct mq_namespace *mq_ns;
+
  BUG_ON(!old_ns);
- return get_mq_ns(old_ns);
+ get_mq_ns(old_ns);
+
+ if (!(flags & CLONE_NEWMQ))
+  return old_ns;
+
+ mq_ns = clone_mq_ns(old_ns);
+
+ put_mq_ns(old_ns);
+ return mq_ns;
 }

 void free_mq_ns(struct kref *kref)

--

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