Subject: [RFC][PATCH 4/5] mqueue namespace: enable the mqueue namespace
Posted by Dave Hansen on Thu, 10 Jul 2008 22:30:51 GMT
View Forum Message <> Reply to Message

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

Move forward and start using the mqueue namespace.

The single super block mount of the file system is modified to allow
one mount per namespace. This is achieved by storing the namespace
in the super_block s_fs_info attribute.

Changes since v5 Feb 28, 2008:

 * fix race on sb->s_fs_info when the namespace is freed

Changes since v4:

 * check mq_ns validity when the message queue is accessed
   through a user mount and eventually return -EACCES if
   mq_ns is bogus

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
---

 linux-2.6.git-dave/include/linux/mq_namespace.h | 2 +
 linux-2.6.git-dave/ipc/mq_namespace.c | 10 +++++-
 linux-2.6.git-dave/ipc/mqueue.c | 37 ++++++++++++++------
 3 files changed, 40 insertions(+), 9 deletions(-)

diff -puN include/linux/mq_namespace.h~mq_namespace-use-mq_namespace include/linux/mq_namespace.h
--- linux-2.6.git/include/linux/mq_namespace.h~mq_namespace-use-mq_namespace 2008-06-24 12:03:18.000000000 -0700
+++ linux-2.6.git-dave/include/linux/mq_namespace.h 2008-06-24 12:03:18.000000000 -0700
@@ -3,6 +3,7 @@

 #include <linux/kref.h>
 #include <linux/err.h>
+#include <linux/fs.h>

 struct vfsmount;

@@ -17,6 +18,7 @@ struct mq_namespace {
 };

 extern struct mq_namespace init_mq_ns;
+extern struct file_system_type mqueue_fs_type;

```
 /* default values */
 #define DFLT_QUEUESMAX 256 /* max number of message queues */
diff -puN ipc/mq_namespace.c~mq_namespace-use-mq_namespace ipc/mq_namespace.c
--- linux-2.6.git/ipc/mq_namespace.c~mq_namespace-use-mq_namespace 2008-06-24
12:03:18.000000000 -0700
+++ linux-2.6.git-dave/ipc/mq_namespace.c 2008-06-24 12:03:18.000000000 -0700
@@ -13,6 +13,7 @@
 #include <linux/slab.h>
 #include <linux/sched.h>
 #include <linux/err.h>
+#include <linux/mount.h>

 static struct mq_namespace *clone_mq_ns(struct mq_namespace *old_ns)
 {
@@ -27,7 +28,12 @@ static struct mq_namespace *clone_mq_ns(
  mq_ns->queues_max = DFLT_QUEUESMAX;
  mq_ns->msg_max  = DFLT_MSGMAX;
  mq_ns->msgsize_max = DFLT_MSGSIZEMAX;
- mq_ns->mnt  = NULL;
+ mq_ns->mnt  = kern_mount_data(&mqueue_fs_type, mq_ns);
+ if (IS_ERR(mq_ns->mnt)) {
+  void *error = mq_ns->mnt;
+  kfree(mq_ns);
+  return error;
+ }
  return mq_ns;
 }

@@ -53,5 +59,7 @@ void free_mq_ns(struct kref *kref)
  struct mq_namespace *ns;

  ns = container_of(kref, struct mq_namespace, kref);
+ ns->mnt->mnt_sb->s_fs_info = NULL;
+ mntput(ns->mnt);
  kfree(ns);
 }
diff -puN ipc/mqueue.c~mq_namespace-use-mq_namespace ipc/mqueue.c
--- linux-2.6.git/ipc/mqueue.c~mq_namespace-use-mq_namespace 2008-06-24
12:03:18.000000000 -0700
+++ linux-2.6.git-dave/ipc/mqueue.c 2008-06-24 12:03:18.000000000 -0700
@@ -204,7 +204,10 @@ static int mqueue_get_sb(struct file_sys
     int flags, const char *dev_name,
     void *data, struct vfsmount *mnt)
 {
- return get_sb_single(fs_type, flags, data, mqueue_fill_super, mnt);
+ if (!(flags & MS_KERNMOUNT))
+  data = current->nsproxy->mq_ns;
```

```
+
+ return get_sb_single_ns(fs_type, flags, data, mqueue_fill_super, mnt);
 }

 static void init_once(struct kmem_cache *cachep, void *foo)
@@ -235,7 +238,7 @@ static void mqueue_delete_inode(struct i
  struct user_struct *user;
  unsigned long mq_bytes;
  int i;
- struct mq_namespace *mq_ns = &init_mq_ns;
+ struct mq_namespace *mq_ns = inode->i_sb->s_fs_info;

  if (S_ISDIR(inode->i_mode)) {
   clear_inode(inode);
@@ -256,7 +259,8 @@ static void mqueue_delete_inode(struct i
  if (user) {
   spin_lock(&mq_lock);
   user->mq_bytes -= mq_bytes;
-  mq_ns->queues_count--;
+  if (mq_ns)
+   mq_ns->queues_count--;
   spin_unlock(&mq_lock);
   free_uid(user);
  }
@@ -268,7 +272,15 @@ static int mqueue_create(struct inode *d
  struct inode *inode;
  struct mq_attr *attr = dentry->d_fsdata;
  int error;
- struct mq_namespace *mq_ns = &init_mq_ns;
+ struct mq_namespace *mq_ns = dir->i_sb->s_fs_info;
+
+ /*
+  * There is a race on sb->s_fs_info with free_mq_ns() but it
+  * shouldn't be an issue as we are only interested in
+  * current->nsproxy->mq_ns which is valid.
+  */
+ if (mq_ns != current->nsproxy->mq_ns)
+  return -EACCES;

  spin_lock(&mq_lock);
  if (mq_ns->queues_count >= mq_ns->queues_max &&
@@ -301,6 +313,15 @@ out_lock:
 static int mqueue_unlink(struct inode *dir, struct dentry *dentry)
 {
    struct inode *inode = dentry->d_inode;
+ struct mq_namespace *mq_ns = dir->i_sb->s_fs_info;
+
+ /*
```

```
+  * There is a race on sb->s_fs_info with free_mq_ns() but it
+  * shouldn't be an issue as we are only interested in
+  * current->nsproxy->mq_ns which is valid.
+  */
+ if (mq_ns != current->nsproxy->mq_ns)
+  return -EACCES;

  dir->i_ctime = dir->i_mtime = dir->i_atime = CURRENT_TIME;
  dir->i_size -= DIRENT_SIZE;
@@ -670,7 +691,7 @@ asmlinkage long sys_mq_open(const char _
  struct file *filp;
  char *name;
  int fd, error;
- struct mq_namespace *mq_ns = &init_mq_ns;
+ struct mq_namespace *mq_ns = current->nsproxy->mq_ns;

  error = audit_mq_open(oflag, mode, u_attr);
  if (error != 0)
@@ -738,7 +759,7 @@ asmlinkage long sys_mq_unlink(const char
  char *name;
  struct dentry *dentry;
  struct inode *inode = NULL;
- struct mq_namespace *mq_ns = &init_mq_ns;
+ struct mq_namespace *mq_ns = current->nsproxy->mq_ns;

  name = getname(u_name);
  if (IS_ERR(name))
@@ -1203,7 +1224,7 @@ static struct super_operations mqueue_su
  .drop_inode = generic_delete_inode,
 };

-static struct file_system_type mqueue_fs_type = {
+struct file_system_type mqueue_fs_type = {
  .name = "mqueue",
  .get_sb = mqueue_get_sb,
  .kill_sb = kill_litter_super,
@@ -1280,7 +1301,7 @@ static int __init init_mqueue_fs(void)
  if (error)
   goto out_sysctl;

- init_mq_ns.mnt = kern_mount(&mqueue_fs_type);
+ init_mq_ns.mnt = kern_mount_data(&mqueue_fs_type, &init_mq_ns);
  if (IS_ERR(init_mq_ns.mnt)) {
   error = PTR_ERR(init_mq_ns.mnt);
   goto out_filesystem;
_
```

Containers mailing list