
Subject: [patch -mm 1/5] mqueue namespace : add struct mq_namespace
Posted by [Cedric Le Goater](#) on Tue, 02 Oct 2007 08:46:09 GMT
[View Forum Message](#) <> [Reply to Message](#)

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

This patch adds a struct mq_namespace holding the common attributes of the mqueue namespace.

The current code is modified to use the default mqueue namespace object 'init_mq_ns' and to prepare the ground for futur dynamic objects.

Todo:

- use CONFIG_NAMESPACE when next -mm is released

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

```
include/linux/mq_namespace.h | 60 ++++++
ipc/mqueue.c                  | 111 ++++++-----
2 files changed, 130 insertions(+), 41 deletions(-)
```

Index: 2.6.23-rc8-mm2/include/linux/mq_namespace.h

=====

```
--- /dev/null
+++ 2.6.23-rc8-mm2/include/linux/mq_namespace.h
@@ -0,0 +1,60 @@
+#ifndef _LINUX_MQ_NAMESPACE_H
+#define _LINUX_MQ_NAMESPACE_H
+
+#include <linux/kref.h>
+
+struct vfsmount;
+
+struct mq_namespace {
+ struct kref kref;
+ struct vfsmount *mnt;
+
+ unsigned int queues_count;
+ unsigned int queues_max;
+ unsigned int msg_max;
+ unsigned int msgsize_max;
+};
+
+extern struct mq_namespace init_mq_ns;
+
+#ifdef CONFIG_POSIX_MQUEUE
+
```

```

+ #define INIT_MQ_NS(ns) .ns = &init_mq_ns,
+
+ static inline struct mq_namespace *get_mq_ns(struct mq_namespace *ns)
+ {
+     if (ns)
+         kref_get(&ns->kref);
+     return ns;
+ }
+
+ extern struct mq_namespace *copy_mq_ns(unsigned long flags,
+     struct mq_namespace *old_ns);
+ extern void free_mq_ns(struct kref *kref);
+
+ static inline void put_mq_ns(struct mq_namespace *ns)
+ {
+     if (ns)
+         kref_put(&ns->kref, free_mq_ns);
+ }
+
+ #else
+
+ #define INIT_MQ_NS(ns)
+
+ static inline struct mq_namespace *get_mq_ns(struct mq_namespace *ns)
+ {
+     return ns;
+ }
+
+ static inline struct mq_namespace *copy_mq_ns(unsigned long flags,
+     struct mq_namespace *old_ns)
+ {
+     return old_ns;
+ }
+
+ static inline void put_mq_ns(struct mq_namespace *ns) { }
+
+ #endif /* CONFIG_POSIX_MQUEUE */
+
+ #endif /* _LINUX_MQ_H */
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
@@ -31,6 +31,7 @@
#include <linux/mutex.h>
#include <linux/nsproxy.h>
#include <linux/pid.h>
#include <linux/mq_namespace.h>

```

```

#include <net/sock.h>
#include "util.h"
@@ -87,12 +88,18 @@ static void remove_notification(struct m

static spinlock_t mq_lock;
static struct kmem_cache *mqueue_inode_cachep;
-static struct vfsmount *mqueue_mnt;

-static unsigned int queues_count;
-static unsigned int queues_max = DFLT_QUEUESMAX;
-static unsigned int msg_max = DFLT_MSGMAX;
-static unsigned int msgsize_max = DFLT_MSGSIZEMAX;
+struct mq_namespace init_mq_ns = {
+ .kref = {
+ .refcount = ATOMIC_INIT(2),
+ },
+ .mnt = NULL,
+ .queues_count = 0,
+ .queues_max = DFLT_QUEUESMAX,
+ .msg_max = DFLT_MSGMAX,
+ .msgsize_max = DFLT_MSGSIZEMAX,
+};
+

static struct ctl_table_header * mq_sysctl_table;

@@ -101,6 +108,21 @@ static inline struct mqueue_inode_info *
return container_of(inode, struct mqueue_inode_info, vfs_inode);
}

+struct mq_namespace *copy_mq_ns(unsigned long flags,
+ struct mq_namespace *old_ns)
+{
+ BUG_ON(!old_ns);
+ return get_mq_ns(old_ns);
+}
+
+void free_mq_ns(struct kref *kref)
+{
+ struct mq_namespace *mq_ns;
+
+ mq_ns = container_of(kref, struct mq_namespace, kref);
+ kfree(mq_ns);
+}
+
static struct inode *mqueue_get_inode(struct super_block *sb, int mode,
struct mq_attr *attr)

```

```

{
@@ -235,6 +257,7 @@ 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;

    if (S_ISDIR(inode->i_mode)) {
        clear_inode(inode);
@@ -255,7 +278,7 @@ static void mqueue_delete_inode(struct i
    if (user) {
        spin_lock(&mq_lock);
        user->mq_bytes -= mq_bytes;
- queues_count--;
+ mq_ns->queues_count--;
        spin_unlock(&mq_lock);
        free_uid(user);
    }
@@ -267,20 +290,22 @@ 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;

    spin_lock(&mq_lock);
- if (queues_count >= queues_max && !capable(CAP_SYS_RESOURCE)) {
+ if (mq_ns->queues_count >= mq_ns->queues_max &&
+ !capable(CAP_SYS_RESOURCE)) {
    error = -ENOSPC;
    goto out_lock;
}
- queues_count++;
+ mq_ns->queues_count++;
    spin_unlock(&mq_lock);

    inode = mqueue_get_inode(dir->i_sb, mode, attr);
    if (!inode) {
        error = -ENOMEM;
        spin_lock(&mq_lock);
- queues_count--;
+ mq_ns->queues_count--;
        goto out_lock;
    }

@@ -571,7 +596,7 @@ static void remove_notification(struct m
    info->notify_owner = NULL;
}

```

```

-static int mq_attr_ok(struct mq_attr *attr)
+static int mq_attr_ok(struct mq_namespace *mq_ns, struct mq_attr *attr)
{
    if (attr->mq_maxmsg <= 0 || attr->mq_msgsize <= 0)
        return 0;
@@ -579,8 +604,8 @@ static int mq_attr_ok(struct mq_attr *at
    if (attr->mq_maxmsg > HARD_MSGMAX)
        return 0;
    } else {
-    if (attr->mq_maxmsg > msg_max ||
-    attr->mq_msgsize > msgsize_max)
+    if (attr->mq_maxmsg > mq_ns->msg_max ||
+    attr->mq_msgsize > mq_ns->msgsize_max)
        return 0;
    }
    /* check for overflow */
@@ -596,8 +621,9 @@ static int mq_attr_ok(struct mq_attr *at
/*
 * Invoked when creating a new queue via sys_mq_open
 */
-static struct file *do_create(struct dentry *dir, struct dentry *dentry,
-    int oflag, mode_t mode, struct mq_attr __user *u_attr)
+static struct file *do_create(struct mq_namespace *mq_ns, struct dentry *dir,
+    struct dentry *dentry, int oflag, mode_t mode,
+    struct mq_attr __user *u_attr)
{
    struct mq_attr attr;
    int ret;
@@ -607,7 +633,7 @@ static struct file *do_create(struct den
    if (copy_from_user(&attr, u_attr, sizeof(attr)))
        goto out;
    ret = -EINVAL;
-    if (!mq_attr_ok(&attr))
+    if (!mq_attr_ok(mq_ns, &attr))
        goto out;
    /* store for use during create */
    dentry->d_fsdata = &attr;
@@ -619,34 +645,34 @@ static struct file *do_create(struct den
    if (ret)
        goto out;

-    return dentry_open(dentry, mqueue_mnt, oflag);
+    return dentry_open(dentry, mq_ns->mnt, oflag);

out:
    dput(dentry);
-    mntput(mqueue_mnt);
+    mntput(mq_ns->mnt);

```

```

    return ERR_PTR(ret);
}

/* Opens existing queue */
-static struct file *do_open(struct dentry *dentry, int oflag)
+static struct file *do_open(struct mq_namespace *mq_ns, struct dentry *dentry,
+ int oflag)
{
    static int oflag2acc[O_ACCMODE] = { MAY_READ, MAY_WRITE,
        MAY_READ | MAY_WRITE };

    if ((oflag & O_ACCMODE) == (O_RDWR | O_WRONLY)) {
        dput(dentry);
    - mntput(mqueue_mnt);
    + mntput(mq_ns->mnt);
        return ERR_PTR(-EINVAL);
    }

    if (permission(dentry->d_inode, oflag2acc[oflag & O_ACCMODE], NULL)) {
        dput(dentry);
    - mntput(mqueue_mnt);
    + mntput(mq_ns->mnt);
        return ERR_PTR(-EACCES);
    }

    - return dentry_open(dentry, mqueue_mnt, oflag);
    + return dentry_open(dentry, mq_ns->mnt, oflag);
}

asmlinkage long sys_mq_open(const char __user *u_name, int oflag, mode_t mode,
@@ -655,6 +682,7 @@ asmlinkage long sys_mq_open(const char _
    struct file *filp;
    char *name;
    int fd, error;
+ struct mq_namespace *mq_ns = &init_mq_ns;

    error = audit_mq_open(oflag, mode, u_attr);
    if (error != 0)
@@ -667,13 +695,13 @@ asmlinkage long sys_mq_open(const char _
    if (fd < 0)
        goto out_putname;

    - mutex_lock(&mqueue_mnt->mnt_root->d_inode->i_mutex);
    - dentry = lookup_one_len(name, mqueue_mnt->mnt_root, strlen(name));
    + mutex_lock(&mq_ns->mnt->mnt_root->d_inode->i_mutex);
    + dentry = lookup_one_len(name, mq_ns->mnt->mnt_root, strlen(name));
    if (IS_ERR(dentry)) {
        error = PTR_ERR(dentry);

```

```

    goto out_err;
}
- mntget(mqueue_mnt);
+ mntget(mq_ns->mnt);

if (oflag & O_CREAT) {
    if (dentry->d_inode) { /* entry already exists */
@@ -681,12 +709,12 @@ asmlinkage long sys_mq_open(const char _
        error = -EEXIST;
        if (oflag & O_EXCL)
            goto out;
-    filp = do_open(dentry, oflag);
+    filp = do_open(mq_ns, dentry, oflag);
    } else {
-    error = mnt_want_write(mqueue_mnt);
+    error = mnt_want_write(mq_ns->mnt);
        if (error)
            goto out;
-    filp = do_create(mqueue_mnt->mnt_root, dentry,
+    filp = do_create(mq_ns, mq_ns->mnt->mnt_root, dentry,
            oflag, mode, u_attr);
    }
} else {
@@ -694,7 +722,7 @@ asmlinkage long sys_mq_open(const char _
    if (!dentry->d_inode)
        goto out;
    audit_inode(name, dentry);
-    filp = do_open(dentry, oflag);
+    filp = do_open(mq_ns, dentry, oflag);
}

if (IS_ERR(filp)) {
@@ -708,13 +736,13 @@ asmlinkage long sys_mq_open(const char _

out:
    dput(dentry);
- mntput(mqueue_mnt);
+ mntput(mq_ns->mnt);
out_putfd:
    put_unused_fd(fd);
out_err:
    fd = error;
out_upsem:
- mutex_unlock(&mqueue_mnt->mnt_root->d_inode->i_mutex);
+ mutex_unlock(&mq_ns->mnt->mnt_root->d_inode->i_mutex);
out_putname:
    putname(name);
    return fd;

```

```

@@ -726,14 +754,15 @@ asmlinkage long sys_mq_unlink(const char
    char *name;
    struct dentry *dentry;
    struct inode *inode = NULL;
+ struct mq_namespace *mq_ns = &init_mq_ns;

    name = getname(u_name);
    if (IS_ERR(name))
        return PTR_ERR(name);

- mutex_lock_nested(&mqueue_mnt->mnt_root->d_inode->i_mutex,
+ mutex_lock_nested(&mq_ns->mnt->mnt_root->d_inode->i_mutex,
    I_MUTEX_PARENT);
- dentry = lookup_one_len(name, mqueue_mnt->mnt_root, strlen(name));
+ dentry = lookup_one_len(name, mq_ns->mnt->mnt_root, strlen(name));
    if (IS_ERR(dentry)) {
        err = PTR_ERR(dentry);
        goto out_unlock;
@@ -747,16 +776,16 @@ asmlinkage long sys_mq_unlink(const char
    inode = dentry->d_inode;
    if (inode)
        atomic_inc(&inode->i_count);
- err = mnt_want_write(mqueue_mnt);
+ err = mnt_want_write(mq_ns->mnt);
    if (err)
        goto out_err;
    err = vfs_unlink(dentry->d_parent->d_inode, dentry);
- mnt_drop_write(mqueue_mnt);
+ mnt_drop_write(mq_ns->mnt);
out_err:
    dput(dentry);

out_unlock:
- mutex_unlock(&mqueue_mnt->mnt_root->d_inode->i_mutex);
+ mutex_unlock(&mq_ns->mnt->mnt_root->d_inode->i_mutex);
    putname(name);
    if (inode)
        iput(inode);
@@ -1201,14 +1230,14 @@ static int msg_maxsize_limit_max = INT_M
static ctl_table mq_sysctls[] = {
    {
        .procname = "queues_max",
- .data = &queues_max,
+ .data = &init_mq_ns.queues_max,
        .maxlen = sizeof(int),
        .mode = 0644,
        .proc_handler = &proc_dointvec,
    },

```

```

{
    .procname = "msg_max",
- .data = &msg_max,
+ .data = &init_mq_ns.msg_max,
    .maxlen = sizeof(int),
    .mode = 0644,
    .proc_handler = &proc_dointvec_minmax,
@@ -1217,7 +1246,7 @@ static ctl_table mq_sysctls[] = {
},
{
    .procname = "msgsize_max",
- .data = &msgsize_max,
+ .data = &init_mq_ns.msgsize_max,
    .maxlen = sizeof(int),
    .mode = 0644,
    .proc_handler = &proc_dointvec_minmax,
@@ -1263,13 +1292,13 @@ static int __init init_mqueue_fs(void)
if (error)
    goto out_sysctl;

- if (IS_ERR(mqueue_mnt = kern_mount(&mqueue_fs_type))) {
- error = PTR_ERR(mqueue_mnt);
+ init_mq_ns.mnt = kern_mount(&mqueue_fs_type);
+ if (IS_ERR(init_mq_ns.mnt)) {
+ error = PTR_ERR(init_mq_ns.mnt);
    goto out_filesystem;
}

/* internal initialization - not common for vfs */
- queues_count = 0;
spin_lock_init(&mq_lock);

return 0;

--

```

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