
Subject: [RFC][PATCH 1/5] mqueue namespace: add struct mq_namespace
Posted by [Dave Hansen](#) on Thu, 10 Jul 2008 22:30:48 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.

A new option CONFIG_MQ_NS protects configuration not using namespaces.

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

```
linux-2.6.git-dave/include/linux/mq_namespace.h | 67 ++++++++
linux-2.6.git-dave/init/Kconfig                 |  9 +
linux-2.6.git-dave/ipc/Makefile                  |  1
linux-2.6.git-dave/ipc/mq_namespace.c           | 23 +++++
linux-2.6.git-dave/ipc/mqueue.c                  | 109 ++++++++-----
5 files changed, 158 insertions(+), 51 deletions(-)
```

```
diff -puN /dev/null include/linux/mq_namespace.h
--- /dev/null 2007-04-11 11:48:27.000000000 -0700
+++ linux-2.6.git-dave/include/linux/mq_namespace.h 2008-06-24 12:03:16.000000000 -0700
@@ -0,0 +1,67 @@
+#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;
+
```

```

+/* default values */
+#define DFLT_QUEUESMAX 256 /* max number of message queues */
+#define DFLT_MSGMAX 10 /* max number of messages in each queue */
+#define HARD_MSGMAX (131072/sizeof(void *))
+#define DFLT_MSGSIZEMAX 8192 /* max message size */
+
+#ifndef CONFIG_POSIX_MQUEUE
+#define INIT_MQ_NS(ns) .ns = &init_mq_ns,
+#else
+#define INIT_MQ_NS(ns)
+#endif
+
+#if defined(CONFIG_POSIX_MQUEUE) && defined(CONFIG_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 clone_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
+
+static inline struct mq_namespace *get_mq_ns(struct mq_namespace *ns)
+{
+ return ns;
+}
+
+static inline struct mq_namespace *copy_mq_ns(unsigned long clone_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 */

```

```
diff -puN init/Kconfig~mq_namespace-add-mq_namespace init/Kconfig
--- linux-2.6.git/init/Kconfig~mq_namespace-add-mq_namespace 2008-06-24
12:03:16.000000000 -0700
+++ linux-2.6.git-dave/init/Kconfig 2008-06-24 12:03:16.000000000 -0700
@@ -493,6 +493,15 @@ config PID_NS
    Unless you want to work with an experimental feature
    say N here.
```

```
+config MQ_NS
+    bool "POSIX Message Queues namespace"
+    depends on NAMESPACES && POSIX_MQUEUE
+    help
+    Support for POSIX Message Queues namespaces. This allows
+    having different POSIX Message Queues filesystems containing
+    message queues with the same name. Yet another a building
+    block of containers.
```

```
+
+config BLK_DEV_INITRD
+    bool "Initial RAM filesystem and RAM disk (initramfs/initrd) support"
+    depends on BROKEN || !FRV
```

```
diff -puN ipc/Makefile~mq_namespace-add-mq_namespace ipc/Makefile
--- linux-2.6.git/ipc/Makefile~mq_namespace-add-mq_namespace 2008-06-24
12:03:16.000000000 -0700
+++ linux-2.6.git-dave/ipc/Makefile 2008-06-24 12:03:16.000000000 -0700
@@ -8,4 +8,5 @@ obj-$(CONFIG_SYSVIPC_SYSCTL) += ipc_sysc
obj_mq-$(CONFIG_COMPAT) += compat_mq.o
obj-$(CONFIG_POSIX_MQUEUE) += mqueue.o msgutil.o $(obj_mq-y)
obj-$(CONFIG_IPC_NS) += namespace.o
+obj-$(CONFIG_MQ_NS) += mq_namespace.o
```

```
diff -puN /dev/null ipc/mq_namespace.c
--- /dev/null 2007-04-11 11:48:27.000000000 -0700
+++ linux-2.6.git-dave/ipc/mq_namespace.c 2008-06-24 12:03:16.000000000 -0700
@@ -0,0 +1,23 @@
+/*
+ * Copyright (C) 2007 IBM Corporation
+ *
+ * Author: Cedric Le Goater <clg@fr.ibm.com>
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License as
+ * published by the Free Software Foundation, version 2 of the
+ * License.
+ */
+
+#include <linux/mq_namespace.h>
+
+struct mq_namespace *copy_mq_ns(unsigned long clone_flags,
```

```

+ struct mq_namespace *old_ns)
+{
+ BUG_ON(!old_ns);
+ return get_mq_ns(old_ns);
+}
+
+void free_mq_ns(struct kref *kref)
+{
+}
diff -puN ipc/mqueue.c~mq_namespace-add-mq_namespace ipc/mqueue.c
--- linux-2.6.git/ipc/mqueue.c~mq_namespace-add-mq_namespace 2008-06-24
12:03:16.000000000 -0700
+++ linux-2.6.git-dave/ipc/mqueue.c 2008-06-24 12:03:16.000000000 -0700
@@ -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"
@@ -46,13 +47,6 @@
#define STATE_PENDING 1
#define STATE_READY 2

/* default values */
#define DFLT_QUEUESMAX 256 /* max number of message queues */
#define DFLT_MSGMAX 10 /* max number of messages in each queue */
#define HARD_MSGMAX (131072/sizeof(void*))
#define DFLT_MSGSIZEMAX 8192 /* max message size */
-
-
struct ext_wait_queue { /* queue of sleeping tasks */
    struct task_struct *task;
    struct list_head list;
@@ -87,12 +81,18 @@ 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;

@@ -235,6 +235,7 @@ 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 +256,7 @@ 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 +268,22 @@ 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;
}

@@ -569,7 +572,7 @@ 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;
@@ -577,8 +580,8 @@ 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 */
@@ -594,8 +597,9 @@ 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;
    struct file *result;
@@ -606,14 +610,14 @@ 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;
}

```

```

mode &= ~current->fs->umask;
- ret = mnt_want_write(mqueue_mnt);
+ ret = mnt_want_write(mq_ns->mnt);
if (ret)
    goto out;
ret = vfs_create(dir->d_inode, dentry, mode, NULL);
@@ -621,41 +625,42 @@ static struct file *do_create(struct den
if (ret)
    goto out_drop_write;

- result = dentry_open(dentry, mqueue_mnt, oflag);
+ result = dentry_open(dentry, mq_ns->mnt, oflag);
/*
 * dentry_open() took a persistent mnt_want_write(),
 * so we can now drop this one.
 */
- mnt_drop_write(mqueue_mnt);
+ mnt_drop_write(mq_ns->mnt);
return result;

out_drop_write:
- mnt_drop_write(mqueue_mnt);
+ mnt_drop_write(mq_ns->mnt);
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,
+ 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,
@@ -665,6 +670,7 @@ 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)
@@ -677,13 +683,13 @@ 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 */
@@ -691,9 +697,9 @@ 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 {
-        filp = do_create(mqueue_mnt->mnt_root, dentry,
+        filp = do_create(mq_ns, mq_ns->mnt->mnt_root, dentry,
            oflag, mode, u_attr);
        }
    } else {

```

```

@@ -701,7 +707,7 @@ 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)) {
@@ -714,13 +720,13 @@ 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;
@@ -732,14 +738,15 @@ 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;
@@ -753,16 +760,16 @@ 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);
@@ -1211,14 +1218,14 @@ 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,
@@ -1227,7 +1234,7 @@ 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,
@@ -1273,13 +1280,13 @@ 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>
