Subject: Re: pspace name
Posted by Cedric Le Goater on Thu, 07 Sep 2006 21:16:58 GMT
View Forum Message <> Reply to Message

Kirill Korotaev wrote:
> Cedric Le Goater wrote:
>> all,
>>
>> 'pspace' sounds wrong when you know about the other namespaces :
>>
>> struct nsproxy {
>>   atomic_t count;
>>   spinlock_t nslock;
>>   struct uts_namespace *uts_ns;
>>   struct ipc_namespace *ipc_ns;
>>   struct user_namespace *user_ns;
>>   struct namespace *namespace;
>> };
>>
>> 'proc_namespace' might be confusing, what about 'task_namespace' ?
> yes, I also wanted to point to this, but probably missed in a hurry.
> task_ns/task_namespace looks fine, doesn't it?
>
>> 'namespace' should probably be renamed to something like
>> 'mnt_namespace' ?
> struct: mnt_namespace
> fields: mnt_ns
>
> is the patch below ok for you?

yep.

touch_mnt_namespace() and the namespace_sem could be renamed but they are
private to namespace.c, so no big issue. here's a port of your patchset for
2.6.18-rc6 which could be used before -mm ?

thanks,

C.

this patch renames 'struct namespace' to 'struct mnt_namespace' and
'namespace' variables to 'mnt_ns' to avoid confusion with other
namespaces being developped for the containers.

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

Cc: containers@lists.osdl.org

---
 fs/afs/mntpt.c | 2
 fs/namespace.c | 112
++++++++++++++++++++----------------
 fs/pnode.c | 2
 fs/pnode.h | 2
 fs/proc/base.c | 36 ++++++-------
 fs/reiserfs/super.c | 2
 include/linux/mnt_namespace.h | 41 ++++++++++++++++
 include/linux/mount.h | 4 -
 include/linux/namespace.h | 44 ----------------
 include/linux/sched.h | 6 +-
 kernel/exit.c | 10 +--
 kernel/fork.c | 32 ++++++------
 kernel/kmod.c | 2
 13 files changed, 147 insertions(+), 148 deletions(-)

Index: 2.6.18-rc6/fs/afs/mntpt.c
===
--- 2.6.18-rc6.orig/fs/afs/mntpt.c
+++ 2.6.18-rc6/fs/afs/mntpt.c
@@ -18,7 +18,7 @@
 #include <linux/pagemap.h>
 #include <linux/mount.h>
 #include <linux/namei.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include "super.h"
 #include "cell.h"
 #include "volume.h"
Index: 2.6.18-rc6/fs/namespace.c
===
--- 2.6.18-rc6.orig/fs/namespace.c
+++ 2.6.18-rc6/fs/namespace.c
@@ -18,7 +18,7 @@
 #include <linux/capability.h>
 #include <linux/module.h>
 #include <linux/seq_file.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/namei.h>
 #include <linux/security.h>
 #include <linux/mount.h>
@@ -141,10 +141,10 @@ struct vfsmount *lookup_mnt(struct vfsmo

 static inline int check_mnt(struct vfsmount *mnt)

```
 {
- return mnt->mnt_namespace == current->namespace;
+ return mnt->mnt_ns == current->mnt_ns;
 }

-static void touch_namespace(struct namespace *ns)
+static void touch_mnt_namespace(struct mnt_namespace *ns)
 {
  if (ns) {
   ns->event = ++event;
@@ -152,7 +152,7 @@ static void touch_namespace(struct names
  }
 }

-static void __touch_namespace(struct namespace *ns)
+static void __touch_mnt_namespace(struct mnt_namespace *ns)
 {
  if (ns && ns->event != event) {
   ns->event = event;
@@ -195,19 +195,19 @@ static void commit_tree(struct vfsmount
  struct vfsmount *parent = mnt->mnt_parent;
  struct vfsmount *m;
  LIST_HEAD(head);
- struct namespace *n = parent->mnt_namespace;
+ struct mnt_namespace *n = parent->mnt_ns;

  BUG_ON(parent == mnt);

  list_add_tail(&head, &mnt->mnt_list);
  list_for_each_entry(m, &head, mnt_list)
-  m->mnt_namespace = n;
+  m->mnt_ns = n;
  list_splice(&head, n->list.prev);

  list_add_tail(&mnt->mnt_hash, mount_hashtable +
     hash(parent, mnt->mnt_mountpoint));
  list_add_tail(&mnt->mnt_child, &parent->mnt_mounts);
- touch_namespace(n);
+ touch_mnt_namespace(n);
 }

 static struct vfsmount *next_mnt(struct vfsmount *p, struct vfsmount *root)
@@ -328,7 +328,7 @@ EXPORT_SYMBOL(mnt_unpin);
 /* iterator */
 static void *m_start(struct seq_file *m, loff_t *pos)
 {
- struct namespace *n = m->private;
+ struct mnt_namespace *n = m->private;
```

```
  struct list_head *p;
  loff_t l = *pos;

@@ -341,7 +341,7 @@ static void *m_start(struct seq_file *m,

 static void *m_next(struct seq_file *m, void *v, loff_t *pos)
 {
- struct namespace *n = m->private;
+ struct mnt_namespace *n = m->private;
  struct list_head *p = ((struct vfsmount *)v)->mnt_list.next;
  (*pos)++;
  return p == &n->list ? NULL : list_entry(p, struct vfsmount, mnt_list);
@@ -534,8 +534,8 @@ void umount_tree(struct vfsmount *mnt, i
  list_for_each_entry(p, kill, mnt_hash) {
   list_del_init(&p->mnt_expire);
   list_del_init(&p->mnt_list);
-  __touch_namespace(p->mnt_namespace);
-  p->mnt_namespace = NULL;
+  __touch_mnt_namespace(p->mnt_ns);
+  p->mnt_ns = NULL;
   list_del_init(&p->mnt_child);
   if (p->mnt_parent != p)
    p->mnt_mountpoint->d_mounted--;
@@ -838,7 +838,7 @@ static int attach_recursive_mnt(struct v
  if (parent_nd) {
   detach_mnt(source_mnt, parent_nd);
   attach_mnt(source_mnt, nd);
-  touch_namespace(current->namespace);
+  touch_mnt_namespace(current->mnt_ns);
  } else {
   mnt_set_mountpoint(dest_mnt, dest_dentry, source_mnt);
   commit_tree(source_mnt);
@@ -1153,9 +1153,9 @@ static void expire_mount(struct vfsmount
   */
  if (!propagate_mount_busy(mnt, 2)) {
   /* delete from the namespace */
-  touch_namespace(mnt->mnt_namespace);
+  touch_mnt_namespace(mnt->mnt_ns);
   list_del_init(&mnt->mnt_list);
-  mnt->mnt_namespace = NULL;
+  mnt->mnt_ns = NULL;
   umount_tree(mnt, 1, umounts);
   spin_unlock(&vfsmount_lock);
  } else {
@@ -1176,7 +1176,7 @@ static void expire_mount(struct vfsmount
  */
 static void expire_mount_list(struct list_head *graveyard, struct
list_head *mounts)
```

```
 {
- struct namespace *namespace;
+ struct mnt_namespace *ns;
  struct vfsmount *mnt;

  while (!list_empty(graveyard)) {
@@ -1186,10 +1186,10 @@ static void expire_mount_list(struct lis

   /* don't do anything if the namespace is dead - all the
    * vfsmounts from it are going away anyway */
-  namespace = mnt->mnt_namespace;
-  if (!namespace || !namespace->root)
+  ns = mnt->mnt_ns;
+  if (!ns || !ns->root)
    continue;
-  get_namespace(namespace);
+  get_mnt_ns(ns);

   spin_unlock(&vfsmount_lock);
   down_write(&namespace_sem);
@@ -1197,7 +1197,7 @@ static void expire_mount_list(struct lis
   up_write(&namespace_sem);
   release_mounts(&umounts);
   mntput(mnt);
-  put_namespace(namespace);
+  put_mnt_ns(ns);
   spin_lock(&vfsmount_lock);
  }
 }
@@ -1447,14 +1447,14 @@ dput_out:
  * Allocate a new namespace structure and populate it with contents
  * copied from the namespace of the passed in task structure.
  */
-struct namespace *dup_namespace(struct task_struct *tsk, struct fs_struct *fs)
+struct mnt_namespace *dup_mnt_ns(struct task_struct *tsk, struct fs_struct
*fs)
 {
- struct namespace *namespace = tsk->namespace;
- struct namespace *new_ns;
+ struct mnt_namespace *mnt_ns = tsk->mnt_ns;
+ struct mnt_namespace *new_ns;
  struct vfsmount *rootmnt = NULL, *pwdmnt = NULL, *altrootmnt = NULL;
  struct vfsmount *p, *q;

- new_ns = kmalloc(sizeof(struct namespace), GFP_KERNEL);
+ new_ns = kmalloc(sizeof(struct mnt_namespace), GFP_KERNEL);
  if (!new_ns)
   return NULL;
```

```
@@ -1465,7 +1465,7 @@ struct namespace *dup_namespace(struct t

  down_write(&namespace_sem);
  /* First pass: copy the tree topology */
- new_ns->root = copy_tree(namespace->root, namespace->root->mnt_root,
+ new_ns->root = copy_tree(mnt_ns->root, mnt_ns->root->mnt_root,
      CL_COPY_ALL | CL_EXPIRE);
  if (!new_ns->root) {
   up_write(&namespace_sem);
@@ -1481,10 +1481,10 @@ struct namespace *dup_namespace(struct t
   * as belonging to new namespace.  We have already acquired a private
   * fs_struct, so tsk->fs->lock is not needed.
   */
- p = namespace->root;
+ p = mnt_ns->root;
  q = new_ns->root;
  while (p) {
-  q->mnt_namespace = new_ns;
+  q->mnt_ns = new_ns;
   if (fs) {
    if (p == fs->rootmnt) {
     rootmnt = p;
@@ -1499,7 +1499,7 @@ struct namespace *dup_namespace(struct t
     fs->altrootmnt = mntget(q);
    }
   }
-  p = next_mnt(p, namespace->root);
+  p = next_mnt(p, mnt_ns->root);
   q = next_mnt(q, new_ns->root);
  }
  up_write(&namespace_sem);
@@ -1514,18 +1514,18 @@ struct namespace *dup_namespace(struct t
  return new_ns;
 }

-int copy_namespace(int flags, struct task_struct *tsk)
+int copy_mnt_ns(int flags, struct task_struct *tsk)
 {
- struct namespace *namespace = tsk->namespace;
- struct namespace *new_ns;
+  struct mnt_namespace *mnt_ns = tsk->mnt_ns;
+  struct mnt_namespace *new_ns;
  int err = 0;

- if (!namespace)
+ if (!mnt_ns)
   return 0;
```

```
- get_namespace(namespace);
+ get_mnt_ns(mnt_ns);

- if (!(flags & CLONE_NEWNS))
+ if (!(flags & CLONE_MNTNS))
   return 0;

  if (!capable(CAP_SYS_ADMIN)) {
@@ -1533,16 +1533,16 @@ int copy_namespace(int flags, struct tas
   goto out;
  }

- new_ns = dup_namespace(tsk, tsk->fs);
+ new_ns = dup_mnt_ns(tsk, tsk->fs);
  if (!new_ns) {
   err = -ENOMEM;
   goto out;
  }

- tsk->namespace = new_ns;
+ tsk->mnt_ns = new_ns;

 out:
- put_namespace(namespace);
+ put_mnt_ns(mnt_ns);
  return err;
 }

@@ -1762,7 +1762,7 @@ asmlinkage long sys_pivot_root(const cha
  detach_mnt(user_nd.mnt, &root_parent);
  attach_mnt(user_nd.mnt, &old_nd);     /* mount old root on put_old */
  attach_mnt(new_nd.mnt, &root_parent); /* mount new_root on / */
- touch_namespace(current->namespace);
+ touch_mnt_namespace(current->mnt_ns);
  spin_unlock(&vfsmount_lock);
  chroot_fs_refs(&user_nd, &new_nd);
  security_sb_post_pivotroot(&user_nd, &new_nd);
@@ -1787,33 +1787,33 @@ out3:
 static void __init init_mount_tree(void)
 {
  struct vfsmount *mnt;
- struct namespace *namespace;
+ struct mnt_namespace *mnt_ns;
  struct task_struct *g, *p;

  mnt = do_kern_mount("rootfs", 0, "rootfs", NULL);
  if (IS_ERR(mnt))
```

```
   panic("Can't create rootfs");
- namespace = kmalloc(sizeof(*namespace), GFP_KERNEL);
- if (!namespace)
+ mnt_ns = kmalloc(sizeof(*mnt_ns), GFP_KERNEL);
+ if (!mnt_ns)
   panic("Can't allocate initial namespace");
- atomic_set(&namespace->count, 1);
- INIT_LIST_HEAD(&namespace->list);
- init_waitqueue_head(&namespace->poll);
- namespace->event = 0;
- list_add(&mnt->mnt_list, &namespace->list);
- namespace->root = mnt;
- mnt->mnt_namespace = namespace;
+ atomic_set(&mnt_ns->count, 1);
+ INIT_LIST_HEAD(&mnt_ns->list);
+ init_waitqueue_head(&mnt_ns->poll);
+ mnt_ns->event = 0;
+ list_add(&mnt->mnt_list, &mnt_ns->list);
+ mnt_ns->root = mnt;
+ mnt->mnt_ns = mnt_ns;

- init_task.namespace = namespace;
+ init_task.mnt_ns = mnt_ns;
  read_lock(&tasklist_lock);
  do_each_thread(g, p) {
-  get_namespace(namespace);
-  p->namespace = namespace;
+  get_mnt_ns(mnt_ns);
+  p->mnt_ns = mnt_ns;
  } while_each_thread(g, p);
  read_unlock(&tasklist_lock);

- set_fs_pwd(current->fs, namespace->root, namespace->root->mnt_root);
- set_fs_root(current->fs, namespace->root, namespace->root->mnt_root);
+ set_fs_pwd(current->fs, mnt_ns->root, mnt_ns->root->mnt_root);
+ set_fs_root(current->fs, mnt_ns->root, mnt_ns->root->mnt_root);
 }

 void __init mnt_init(unsigned long mempages)
@@ -1867,11 +1867,11 @@ void __init mnt_init(unsigned long mempa
  init_mount_tree();
 }

-void __put_namespace(struct namespace *namespace)
+void __put_mnt_ns(struct mnt_namespace *ns)
 {
- struct vfsmount *root = namespace->root;
+ struct vfsmount *root = ns->root;
```

```
  LIST_HEAD(umount_list);
- namespace->root = NULL;
+ ns->root = NULL;
  spin_unlock(&vfsmount_lock);
  down_write(&namespace_sem);
  spin_lock(&vfsmount_lock);
@@ -1879,5 +1879,5 @@ void __put_namespace(struct namespace *n
  spin_unlock(&vfsmount_lock);
  up_write(&namespace_sem);
  release_mounts(&umount_list);
- kfree(namespace);
+ kfree(ns);
 }
Index: 2.6.18-rc6/fs/pnode.c
===================================================================
--- 2.6.18-rc6.orig/fs/pnode.c
+++ 2.6.18-rc6/fs/pnode.c
@@ -6,7 +6,7 @@
  * Author : Ram Pai (linuxram@us.ibm.com)
  *
  */
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/mount.h>
 #include <linux/fs.h>
 #include "pnode.h"
Index: 2.6.18-rc6/fs/pnode.h
===================================================================
--- 2.6.18-rc6.orig/fs/pnode.h
+++ 2.6.18-rc6/fs/pnode.h
@@ -13,7 +13,7 @@

 #define IS_MNT_SHARED(mnt) (mnt->mnt_flags & MNT_SHARED)
 #define IS_MNT_SLAVE(mnt) (mnt->mnt_master)
-#define IS_MNT_NEW(mnt)  (!mnt->mnt_namespace)
+#define IS_MNT_NEW(mnt)  (!mnt->mnt_ns)
 #define CLEAR_MNT_SHARED(mnt) (mnt->mnt_flags &= ~MNT_SHARED)
 #define IS_MNT_UNBINDABLE(mnt) (mnt->mnt_flags & MNT_UNBINDABLE)

Index: 2.6.18-rc6/fs/proc/base.c
===================================================================
--- 2.6.18-rc6.orig/fs/proc/base.c
+++ 2.6.18-rc6/fs/proc/base.c
@@ -59,7 +59,7 @@
 #include <linux/string.h>
 #include <linux/seq_file.h>
 #include <linux/namei.h>
-#include <linux/namespace.h>
```

```
+#include <linux/mnt_namespace.h>
 #include <linux/mm.h>
 #include <linux/smp_lock.h>
 #include <linux/rcupdate.h>
@@ -581,33 +581,33 @@ struct proc_mounts {
 static int mounts_open(struct inode *inode, struct file *file)
 {
  struct task_struct *task = get_proc_task(inode);
- struct namespace *namespace = NULL;
+ struct mnt_namespace *mnt_ns = NULL;
  struct proc_mounts *p;
  int ret = -EINVAL;

  if (task) {
   task_lock(task);
-  namespace = task->namespace;
-  if (namespace)
-   get_namespace(namespace);
+  mnt_ns = task->mnt_ns;
+  if (mnt_ns)
+   get_mnt_ns(mnt_ns);
   task_unlock(task);
   put_task_struct(task);
  }

- if (namespace) {
+ if (mnt_ns) {
   ret = -ENOMEM;
   p = kmalloc(sizeof(struct proc_mounts), GFP_KERNEL);
   if (p) {
    file->private_data = &p->m;
    ret = seq_open(file, &mounts_op);
    if (!ret) {
-    p->m.private = namespace;
-    p->event = namespace->event;
+    p->m.private = mnt_ns;
+    p->event = mnt_ns->event;
     return 0;
    }
    kfree(p);
   }
-  put_namespace(namespace);
+  put_mnt_ns(mnt_ns);
  }
  return ret;
 }
@@ -615,15 +615,15 @@ static int mounts_open(struct inode *ino
 static int mounts_release(struct inode *inode, struct file *file)
```

```
 {
  struct seq_file *m = file->private_data;
- struct namespace *namespace = m->private;
- put_namespace(namespace);
+ struct mnt_namespace *ns = m->private;
+ put_mnt_ns(ns);
  return seq_release(inode, file);
 }

 static unsigned mounts_poll(struct file *file, poll_table *wait)
 {
  struct proc_mounts *p = file->private_data;
- struct namespace *ns = p->m.private;
+ struct mnt_namespace *ns = p->m.private;
  unsigned res = 0;

  poll_wait(file, &ns->poll, wait);
@@ -653,20 +653,20 @@ static int mountstats_open(struct inode

  if (!ret) {
   struct seq_file *m = file->private_data;
-  struct namespace *namespace = NULL;
+  struct mnt_namespace *mnt_ns = NULL;
   struct task_struct *task = get_proc_task(inode);

   if (task) {
    task_lock(task);
-   namespace = task->namespace;
-   if (namespace)
-    get_namespace(namespace);
+   mnt_ns = task->mnt_ns;
+   if (mnt_ns)
+    get_mnt_ns(mnt_ns);
    task_unlock(task);
    put_task_struct(task);
   }

-  if (namespace)
-   m->private = namespace;
+  if (mnt_ns)
+   m->private = mnt_ns;
   else {
    seq_release(inode, file);
    ret = -EINVAL;
Index: 2.6.18-rc6/fs/reiserfs/super.c
===================================================================
--- 2.6.18-rc6.orig/fs/reiserfs/super.c
+++ 2.6.18-rc6/fs/reiserfs/super.c
```

```
@@ -23,7 +23,7 @@
 #include <linux/blkdev.h>
 #include <linux/buffer_head.h>
 #include <linux/vfs.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/mount.h>
 #include <linux/namei.h>
 #include <linux/quotaops.h>
Index: 2.6.18-rc6/include/linux/mnt_namespace.h
===================================================================
--- /dev/null
+++ 2.6.18-rc6/include/linux/mnt_namespace.h
@@ -0,0 +1,41 @@
+#ifndef _MNT_NAMESPACE_H_
+#define _MNT_NAMESPACE_H_
+#ifdef __KERNEL__
+
+#include <linux/mount.h>
+#include <linux/sched.h>
+
+struct mnt_namespace {
+ atomic_t  count;
+ struct vfsmount * root;
+ struct list_head list;
+ wait_queue_head_t poll;
+ int event;
+};
+
+extern int copy_mnt_ns(int, struct task_struct *);
+extern void __put_mnt_ns(struct mnt_namespace *ns);
+extern struct mnt_namespace *dup_mnt_ns(struct task_struct *,
+  struct fs_struct *);
+
+static inline void put_mnt_ns(struct mnt_namespace *ns)
+{
+ if (atomic_dec_and_lock(&ns->count, &vfsmount_lock))
+  /* releases vfsmount_lock */
+  __put_mnt_ns(ns);
+}
+
+static inline void exit_mnt_ns(struct task_struct *p)
+{
+ struct mnt_namespace *ns = p->mnt_ns;
+ if (ns)
+  put_mnt_ns(ns);
+}
+
```

```
+static inline void get_mnt_ns(struct mnt_namespace *ns)
+{
+ atomic_inc(&ns->count);
+}
+
+#endif
+#endif
Index: 2.6.18-rc6/include/linux/mount.h
===================================================================
--- 2.6.18-rc6.orig/include/linux/mount.h
+++ 2.6.18-rc6/include/linux/mount.h
@@ -20,7 +20,7 @@
 struct super_block;
 struct vfsmount;
 struct dentry;
-struct namespace;
+struct mnt_namespace;

 #define MNT_NOSUID 0x01
 #define MNT_NODEV 0x02
@@ -52,7 +52,7 @@ struct vfsmount {
  struct list_head mnt_slave_list;/* list of slave mounts */
  struct list_head mnt_slave; /* slave list entry */
  struct vfsmount *mnt_master; /* slave is on master->mnt_slave_list */
- struct namespace *mnt_namespace; /* containing namespace */
+ struct mnt_namespace *mnt_ns; /* containing namespace */
  int mnt_pinned;
 };

Index: 2.6.18-rc6/include/linux/namespace.h
===================================================================
--- 2.6.18-rc6.orig/include/linux/namespace.h
+++ /dev/null
@@ -1,44 +0,0 @@
-#ifndef _NAMESPACE_H_
-#define _NAMESPACE_H_
-#ifdef __KERNEL__
-
-#include <linux/mount.h>
-#include <linux/sched.h>
-
-struct namespace {
- atomic_t  count;
- struct vfsmount * root;
- struct list_head list;
- wait_queue_head_t poll;
- int event;
-};
```

```
-
-extern int copy_namespace(int, struct task_struct *);
-extern void __put_namespace(struct namespace *namespace);
-extern struct namespace *dup_namespace(struct task_struct *, struct
fs_struct *);
-
-static inline void put_namespace(struct namespace *namespace)
-{
- if (atomic_dec_and_lock(&namespace->count, &vfsmount_lock))
-  /* releases vfsmount_lock */
-  __put_namespace(namespace);
-}
-
-static inline void exit_namespace(struct task_struct *p)
-{
- struct namespace *namespace = p->namespace;
- if (namespace) {
-  task_lock(p);
-  p->namespace = NULL;
-  task_unlock(p);
-  put_namespace(namespace);
- }
-}
-
-static inline void get_namespace(struct namespace *namespace)
-{
- atomic_inc(&namespace->count);
-}
-
-#endif
-#endif
Index: 2.6.18-rc6/kernel/exit.c
===================================================================
--- 2.6.18-rc6.orig/kernel/exit.c
+++ 2.6.18-rc6/kernel/exit.c
@@ -13,7 +13,7 @@
 #include <linux/completion.h>
 #include <linux/personality.h>
 #include <linux/tty.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/key.h>
 #include <linux/security.h>
 #include <linux/cpu.h>
@@ -408,9 +408,9 @@ void daemonize(const char *name, ...)
  fs = init_task.fs;
  current->fs = fs;
  atomic_inc(&fs->count);
```

```
- exit_namespace(current);
- current->namespace = init_task.namespace;
- get_namespace(current->namespace);
+ exit_mnt_ns(current);
+ current->mnt_ns = init_task.mnt_ns;
+ get_mnt_ns(current->mnt_ns);
   exit_files(current);
  current->files = init_task.files;
  atomic_inc(&current->files->count);
@@ -916,7 +916,7 @@ fastcall NORET_TYPE void do_exit(long co
  exit_sem(tsk);
  __exit_files(tsk);
  __exit_fs(tsk);
- exit_namespace(tsk);
+ exit_mnt_ns(tsk);
  exit_thread();
  cpuset_exit(tsk);
  exit_keys(tsk);
Index: 2.6.18-rc6/kernel/fork.c
===================================================================
--- 2.6.18-rc6.orig/kernel/fork.c
+++ 2.6.18-rc6/kernel/fork.c
@@ -18,7 +18,7 @@
 #include <linux/module.h>
 #include <linux/vmalloc.h>
 #include <linux/completion.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/personality.h>
 #include <linux/mempolicy.h>
 #include <linux/sem.h>
@@ -948,7 +948,7 @@ static struct task_struct *copy_process(
  int retval;
  struct task_struct *p = NULL;

- if ((clone_flags & (CLONE_NEWNS|CLONE_FS)) == (CLONE_NEWNS|CLONE_FS))
+ if ((clone_flags & (CLONE_MNTNS|CLONE_FS)) == (CLONE_MNTNS|CLONE_FS))
   return ERR_PTR(-EINVAL);

  /*
@@ -1104,7 +1104,7 @@ static struct task_struct *copy_process(
   goto bad_fork_cleanup_signal;
  if ((retval = copy_keys(clone_flags, p)))
   goto bad_fork_cleanup_mm;
- if ((retval = copy_namespace(clone_flags, p)))
+ if ((retval = copy_mnt_ns(clone_flags, p)))
   goto bad_fork_cleanup_keys;
  retval = copy_thread(0, clone_flags, stack_start, stack_size, p, regs);
```

```
  if (retval)
@@ -1252,7 +1252,7 @@ static struct task_struct *copy_process(
  return p;

 bad_fork_cleanup_namespace:
- exit_namespace(p);
+ exit_mnt_ns(p);
 bad_fork_cleanup_keys:
  exit_keys(p);
 bad_fork_cleanup_mm:
@@ -1468,7 +1468,7 @@ static inline void check_unshare_flags(u
  /*
   * If unsharing namespace, must also unshare filesystem information.
   */
- if (*flags_ptr & CLONE_NEWNS)
+ if (*flags_ptr & CLONE_MNTNS)
   *flags_ptr |= CLONE_FS;
 }

@@ -1503,16 +1503,18 @@ static int unshare_fs(unsigned long unsh
 /*
  * Unshare the namespace structure if it is being shared
  */
-static int unshare_namespace(unsigned long unshare_flags, struct namespace
**new_nsp, struct fs_struct *new_fs)
+static int unshare_mnt_namespace(unsigned long unshare_flags,
+     struct mnt_namespace **new_nsp,
+     struct fs_struct *new_fs)
 {
- struct namespace *ns = current->namespace;
+ struct mnt_namespace *ns = current->mnt_ns;

- if ((unshare_flags & CLONE_NEWNS) &&
+ if ((unshare_flags & CLONE_MNTNS) &&
      (ns && atomic_read(&ns->count) > 1)) {
   if (!capable(CAP_SYS_ADMIN))
    return -EPERM;

-  *new_nsp = dup_namespace(current, new_fs ? new_fs : current->fs);
+  *new_nsp = dup_mnt_ns(current, new_fs ? new_fs : current->fs);
   if (!*new_nsp)
    return -ENOMEM;
  }
@@ -1592,7 +1594,7 @@ asmlinkage long sys_unshare(unsigned lon
 {
  int err = 0;
  struct fs_struct *fs, *new_fs = NULL;
- struct namespace *ns, *new_ns = NULL;
```

```
+ struct mnt_namespace *ns, *new_ns = NULL;
  struct sighand_struct *sigh, *new_sigh = NULL;
  struct mm_struct *mm, *new_mm = NULL, *active_mm = NULL;
  struct files_struct *fd, *new_fd = NULL;
@@ -1602,7 +1604,7 @@ asmlinkage long sys_unshare(unsigned lon

  /* Return -EINVAL for all unsupported flags */
  err = -EINVAL;
- if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
+ if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_MNTNS|CLONE_SIGHAND|
     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM))
   goto bad_unshare_out;

@@ -1610,7 +1612,7 @@ asmlinkage long sys_unshare(unsigned lon
   goto bad_unshare_out;
  if ((err = unshare_fs(unshare_flags, &new_fs)))
   goto bad_unshare_cleanup_thread;
- if ((err = unshare_namespace(unshare_flags, &new_ns, new_fs)))
+ if ((err = unshare_mnt_namespace(unshare_flags, &new_ns, new_fs)))
   goto bad_unshare_cleanup_fs;
  if ((err = unshare_sighand(unshare_flags, &new_sigh)))
   goto bad_unshare_cleanup_ns;
@@ -1632,8 +1634,8 @@ asmlinkage long sys_unshare(unsigned lon
   }

   if (new_ns) {
-   ns = current->namespace;
-   current->namespace = new_ns;
+   ns = current->mnt_ns;
+   current->mnt_ns = new_ns;
    new_ns = ns;
   }

@@ -1676,7 +1678,7 @@ bad_unshare_cleanup_sigh:

 bad_unshare_cleanup_ns:
  if (new_ns)
-  put_namespace(new_ns);
+  put_mnt_ns(new_ns);

 bad_unshare_cleanup_fs:
  if (new_fs)
Index: 2.6.18-rc6/kernel/kmod.c
===================================================================
--- 2.6.18-rc6.orig/kernel/kmod.c
+++ 2.6.18-rc6/kernel/kmod.c
@@ -27,7 +27,7 @@
 #include <linux/kmod.h>
```

```
 #include <linux/smp_lock.h>
 #include <linux/slab.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/completion.h>
 #include <linux/file.h>
 #include <linux/workqueue.h>
Index: 2.6.18-rc6/include/linux/sched.h
===================================================================
--- 2.6.18-rc6.orig/include/linux/sched.h
+++ 2.6.18-rc6/include/linux/sched.h
@@ -15,7 +15,7 @@
 #define CLONE_VFORK 0x00004000 /* set if the parent wants the child to
wake it up on mm_release */
 #define CLONE_PARENT 0x00008000 /* set if we want to have the same parent
as the cloner */
 #define CLONE_THREAD 0x00010000 /* Same thread group? */
-#define CLONE_NEWNS 0x00020000 /* New namespace group? */
+#define CLONE_MNTNS 0x00020000 /* New mnt namespace group? */
 #define CLONE_SYSVSEM 0x00040000 /* share system V SEM_UNDO semantics */
 #define CLONE_SETTLS 0x00080000 /* create a new TLS for the child */
 #define CLONE_PARENT_SETTID 0x00100000 /* set the TID in the parent */
@@ -238,7 +238,7 @@ extern signed long schedule_timeout_inte
 extern signed long schedule_timeout_uninterruptible(signed long timeout);
 asmlinkage void schedule(void);

-struct namespace;
+struct mnt_namespace;

 /* Maximum number of active map areas.. This is a random (large) number */
 #define DEFAULT_MAX_MAP_COUNT 65536
@@ -881,7 +881,7 @@ struct task_struct {
 /* open file information */
  struct files_struct *files;
 /* namespace */
- struct namespace *namespace;
+ struct mnt_namespace *mnt_ns;
 /* signal handlers */
  struct signal_struct *signal;
  struct sighand_struct *sighand;
```

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers