

---

Subject: Re: pspace name

Posted by [serue](#) on Thu, 07 Sep 2006 15:41:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

---

Quoting Kirill Korotaev (dev@sw.ru):

> 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?

I still think pid\_ns is more correct, but task\_ns sounds nicer and is at any rate unambiguous.

> > 'namespace' should probably be renamed to something like 'mnt\_namespace' ?

> struct: mnt\_namespace

> fields: mnt\_ns

>

> is the patch below ok for you?

Based on a cursory glance, looks good to me.

-serge

>

> ---

>

> --- ./fs/afs/mntpt.c.mntr 2006-07-14 19:08:29.000000000 +0400

> +++ ./fs/afs/mntpt.c 2006-09-07 18:47:27.000000000 +0400

> @@ -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"
> --- ./fs/namespace.c.mntr 2006-07-14 19:11:05.000000000 +0400
> +++ ./fs/namespace.c 2006-09-07 19:28:05.000000000 +0400
> @@ -20,7 +20,7 @@
> #include <linux/module.h>
> #include <linux/sysfs.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>
> @@ -134,10 +134,10 @@ struct vfsmount *lookup_mnt(struct vfsmo
>
> static inline int check_mnt(struct vfsmount *mnt)
> {
> - return mnt->mnt_namespace == current->nsproxy->namespace;
> + return mnt->mnt_ns == current->nsproxy->mnt_ns;
> }
>
> -static void touch_namespace(struct namespace *ns)
> +static void touch_mnt_namespace(struct mnt_namespace *ns)
> {
> if (ns) {
> ns->event = ++event;
> @@ -145,7 +145,7 @@ 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;
> @@ -188,19 +188,19 @@ 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)
> @@ -321,7 +321,7 @@ 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;
>
> @@ -334,7 +334,7 @@ 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);
> @@ -527,8 +527,8 @@ 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--;
> @@ -831,7 +831,7 @@ static int attach_recursive_mnt(struct v
> if (parent_nd) {
> detach_mnt(source_mnt, parent_nd);
> attach_mnt(source_mnt, nd);
> - touch_namespace(current->nsproxy->namespace);
> + touch_mnt_namespace(current->nsproxy->mnt_ns);
> } else {
> mnt_set_mountpoint(dest_mnt, dest_dentry, source_mnt);
> commit_tree(source_mnt);

```

```

> @@ -1146,9 +1146,9 @@ 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 {
> @@ -1169,7 +1169,7 @@ 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)) {
> @@ -1179,10 +1179,10 @@ 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);
> @@ -1190,7 +1190,7 @@ 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);
> }
> }
> @@ -1440,14 +1440,15 @@ 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->nsproxy->namespace;
> - struct namespace *new_ns;
> + struct mnt_namespace *mnt_ns = tsk->nsproxy->mnt_ns;
> + struct mnt_namespace *new_ns;
>   struct vfsmount *rootmnt = NULL, *pwmnt = 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;
>
> @@ -1458,7 +1459,7 @@ 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);
> @@ -1474,10 +1475,10 @@ 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;
> @@ -1492,7 +1493,7 @@ 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);
> @@ -1507,16 +1508,16 @@ 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->nsproxy->namespace;
> - struct namespace *new_ns;
> + struct mnt_namespace *ns = tsk->nsproxy->mnt_ns;
> + struct mnt_namespace *new_ns;
> int err = 0;
>
> - if (!namespace)
> + if (!ns)
> return 0;
>
> - get_namespace(namespace);
> + get_mnt_ns(ns);
>
> if (!(flags & CLONE_NEWNS))
> return 0;
> @@ -1526,16 +1527,16 @@ 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->nsproxy->namespace = new_ns;
> + tsk->nsproxy->mnt_ns = new_ns;
>
> out:
> - put_namespace(namespace);
> + put_mnt_ns(ns);
> return err;
> }
>
> @@ -1755,7 +1756,7 @@ 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->nsproxy->namespace);
> + touch_mnt_namespace(current->nsproxy->mnt_ns);
> spin_unlock(&vfsmount_lock);

```

```

> chroot_fs_refs(&user_nd, &new_nd);
> security_sb_post_pivotroot(&user_nd, &new_nd);
> @@ -1780,27 +1781,27 @@ out3:
> static void __init init_mount_tree(void)
> {
> struct vfsmount *mnt;
> - struct namespace *namespace;
> + struct mnt_namespace *ns;
>
> 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)
> + ns = kmalloc(sizeof(*ns), GFP_KERNEL);
> + if (!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(&ns->count, 1);
> + INIT_LIST_HEAD(&ns->list);
> + init_waitqueue_head(&ns->poll);
> + ns->event = 0;
> + list_add(&mnt->mnt_list, &ns->list);
> + ns->root = mnt;
> + mnt->mnt_ns = ns;
>
> - init_task.nsproxy->namespace = namespace;
> - get_namespace(namespace);
> + init_task.nsproxy->mnt_ns = ns;
> + get_mnt_ns(ns);
>
> - 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, ns->root, ns->root->mnt_root);
> + set_fs_root(current->fs, ns->root, ns->root->mnt_root);
> }
>
> void __init mnt_init(unsigned long mempages)
> @@ -1861,11 +1862,11 @@ 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);
> @@ -1873,5 +1874,5 @@ void __put_namespace(struct namespace *n
> spin_unlock(&vfsmount_lock);
> up_write(&namespace_sem);
> release_mounts(&umount_list);
> - kfree(namespace);
> + kfree(ns);
> }
> --- ./fs/pnode.c.mntr 2006-07-14 19:08:29.000000000 +0400
> +++ ./fs/pnode.c 2006-09-07 18:47:27.000000000 +0400
> @@ -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"
> --- ./fs/pnode.h.mntr 2006-06-18 05:49:35.000000000 +0400
> +++ ./fs/pnode.h 2006-09-07 18:47:27.000000000 +0400
> @@ -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)
>
> --- ./fs/proc/base.c.mntr 2006-07-14 19:11:05.000000000 +0400
> +++ ./fs/proc/base.c 2006-09-07 18:55:01.000000000 +0400
> @@ -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>
> @@ -561,33 +561,33 @@ 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 *ns = NULL;
>     struct proc_mounts *p;
>     int ret = -EINVAL;
>
>     if (task) {
>         task_lock(task);
> - namespace = task->nsproxy->namespace;
> - if (namespace)
> -     get_namespace(namespace);
> + ns = task->nsproxy->mnt_ns;
> + if (ns)
> +     get_mnt_ns(ns);
>         task_unlock(task);
>         put_task_struct(task);
>     }
>
> - if (namespace) {
> + if (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 = ns;
> +         p->event = ns->event;
>         return 0;
>     }
>     kfree(p);
> }
> - put_namespace(namespace);
> + put_mnt_ns(ns);
> }
> return ret;
> }
> @@ -595,15 +595,15 @@ 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);
> @@ -633,20 +633,20 @@ static int mountstats_open(struct inode
>
> if (!ret) {
> struct seq_file *m = file->private_data;
> - struct namespace *namespace = NULL;
> + struct mnt_namespace *ns = NULL;
> struct task_struct *task = get_proc_task(inode);
>
> if (task) {
> task_lock(task);
> - namespace = task->nsproxy->namespace;
> - if (namespace)
> - get_namespace(namespace);
> + ns = task->nsproxy->mnt_ns;
> + if (ns)
> + get_mnt_ns(ns);
> task_unlock(task);
> put_task_struct(task);
> }
>
> - if (namespace)
> - m->private = namespace;
> + if (ns)
> + m->private = ns;
> else {
> seq_release(inode, file);
> ret = -EINVAL;
> --- ./fs/reiserfs/super.c.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./fs/reiserfs/super.c 2006-09-07 18:47:27.000000000 +0400
> @@ -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>
> --- ./include/linux/init_task.h.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./include/linux/init_task.h 2006-09-07 19:10:28.000000000 +0400
> @@ -76,7 +76,7 @@ extern struct nsproxy init_nsproxy;
> .count = ATOMIC_INIT(1), \
> .nslock = SPIN_LOCK_UNLOCKED, \
> .uts_ns = &init_uts_ns, \
> - .namespace = NULL, \
> + .mnt_ns = NULL, \
> INIT_IPC_NS(ipc_ns) \
> }
>
> --- /dev/null 2003-04-26 02:10:32.000000000 +0400
> +++ ./include/linux/mnt_namespace.h 2006-09-07 18:58:48.000000000 +0400
> @@ -0,0 +1,42 @@
> +#ifndef _NAMESPACE_H_
> +#define _NAMESPACE_H_
> +#ifdef __KERNEL__
> +
> +#include <linux/mount.h>
> +#include <linux/sched.h>
> +#include <linux/nsproxy.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->nsproxy->mnt_ns;
> + if (ns)
> + put_mnt_ns(ns);
> +}
> +
> +static inline void get_mnt_ns(struct mnt_namespace *ns)
> +{
> + atomic_inc(&ns->count);
> +}
> +
> +#endif
> +#endif
> --- ./include/linux/mount.h.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./include/linux/mount.h 2006-09-07 18:47:27.000000000 +0400
> @@ -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;
> };
>
> --- ./include/linux/namespace.h.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./include/linux/namespace.h 2006-09-07 18:57:03.000000000 +0400
> @@ -1,42 +0,0 @@
> -#ifndef _NAMESPACE_H_
> -#define _NAMESPACE_H_
> -#ifdef __KERNEL__
> -
> -#include <linux/mount.h>
> -#include <linux/sched.h>
> -#include <linux/nsproxy.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->nsproxy->namespace;
> - if (namespace) {
> - put_namespace(namespace);
> - }
> -}
> -
> -static inline void get_namespace(struct namespace *namespace)
> -{
> - atomic_inc(&namespace->count);
> -}
> -
> -#endif
> -#endif
> --- ./include/linux/nsproxy.h.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./include/linux/nsproxy.h 2006-09-07 18:59:51.000000000 +0400
> @@ -4,7 +4,7 @@
> #include <linux/spinlock.h>
> #include <linux/sched.h>
>
> -struct namespace;
> +struct mnt_namespace;
> struct uts_namespace;
> struct ipc_namespace;
>
> @@ -25,7 +25,7 @@ struct nsproxy {
> spinlock_t nslock;
> struct uts_namespace *uts_ns;
> struct ipc_namespace *ipc_ns;
> - struct namespace *namespace;
> + struct mnt_namespace *mnt_ns;

```

```

> };
> extern struct nsproxy init_nsproxy;
>
> --- ./kernel/exit.c.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./kernel/exit.c 2006-09-07 18:47:28.000000000 +0400
> @@ -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>
> --- ./kernel/fork.c.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./kernel/fork.c 2006-09-07 19:13:50.000000000 +0400
> @@ -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>
> @@ -1497,17 +1497,18 @@ static int unshare_fs(unsigned long unsh
> }
>
> /*
> - * Unshare the namespace structure if it is being shared
> + * Unshare the mnt_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->nsproxy->namespace;
> + struct mnt_namespace *ns = current->nsproxy->mnt_ns;
>
> if ((unshare_flags & CLONE_NEWNS) && ns) {
> 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;

```

```

> }
> @@ -1597,7 +1598,7 @@ 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;
> @@ -1619,7 +1620,7 @@ 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;
> @@ -1660,8 +1661,8 @@ asmlinkage long sys_unshare(unsigned lon
> }
>
> if (new_ns) {
> - ns = current->nsproxy->namespace;
> - current->nsproxy->namespace = new_ns;
> + ns = current->nsproxy->mnt_ns;
> + current->nsproxy->mnt_ns = new_ns;
> new_ns = ns;
> }
>
> @@ -1728,7 +1729,7 @@ 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)
> --- ./kernel/kmod.c.mntr 2006-07-14 19:08:32.000000000 +0400
> +++ ./kernel/kmod.c 2006-09-07 18:47:28.000000000 +0400
> @@ -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>
> --- ./kernel/nsproxy.c.mntr 2006-07-14 19:11:06.000000000 +0400
> +++ ./kernel/nsproxy.c 2006-09-07 19:16:18.000000000 +0400
> @@ -16,7 +16,7 @@
> #include <linux/module.h>
> #include <linux/version.h>
> #include <linux/nsproxy.h>
> -#include <linux/namespace.h>
> +#include <linux/mnt_namespace.h>
> #include <linux/utsname.h>
>
> static inline void get_nsproxy(struct nsproxy *ns)
> @@ -59,8 +59,8 @@ struct nsproxy *dup_namespaces(struct ns
> struct nsproxy *ns = clone_namespaces(orig);
>
> if (ns) {
> - if (ns->namespace)
> - get_namespace(ns->namespace);
> + if (ns->mnt_ns)
> + get_mnt_ns(ns->mnt_ns);
> if (ns->uts_ns)
> get_uts_ns(ns->uts_ns);
> if (ns->ipc_ns)
> @@ -96,7 +96,7 @@ int copy_namespaces(int flags, struct ta
>
> tsk->nsproxy = new_ns;
>
> - err = copy_namespace(flags, tsk);
> + err = copy_mnt_ns(flags, tsk);
> if (err)
> goto out_ns;
>
> @@ -116,8 +116,8 @@ out_ipc:
> if (new_ns->uts_ns)
> put_uts_ns(new_ns->uts_ns);
> out_uts:
> - if (new_ns->namespace)
> - put_namespace(new_ns->namespace);
> + if (new_ns->mnt_ns)
> + put_mnt_ns(new_ns->mnt_ns);
> out_ns:
> tsk->nsproxy = old_ns;
> put_nsproxy(new_ns);
> @@ -126,8 +126,8 @@ out_ns:
>
> void free_nsproxy(struct nsproxy *ns)
> {

```



```
> - if (ns->namespace)
> - put_namespace(ns->namespace);
> + if (ns->mnt_ns)
> + put_mnt_ns(ns->mnt_ns);
>   if (ns->uts_ns)
>     put_uts_ns(ns->uts_ns);
>   if (ns->ipc_ns)
>


---


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

---

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

---