
Subject: Re: [RFC][PATCH 4/4]: Enable cloning PTY namespaces

Posted by [Pavel Emelianov](#) on Wed, 06 Feb 2008 09:04:15 GMT

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

sukadev@us.ibm.com wrote:

> From: Sukadev Bhattiprolu <sukadev@us.ibm.com>

> Subject: [RFC][PATCH 4/4]: Enable cloning PTY namespaces

>

> Enable cloning PTY namespaces.

>

> TODO:

> This version temporarily uses the clone flag '0x80000000' which

> is unused in mainline atm, but used for CLONE_IO in -mm.

> While we must extend clone() (urgently) to solve this, it hopefully

> does not affect review of the rest of this patchset.

>

> Changelog:

> - Version 0: Based on earlier versions from Serge Hallyn and

> Matt Helsley.

>

> Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>

> ---

> fs/devpts/inode.c | 84 ++++++-----

> include/linux/devpts_fs.h | 52 ++++++

> include/linux/init_task.h | 1

> include/linux/nsproxy.h | 2 +

> include/linux/sched.h | 2 +

> kernel/fork.c | 2 -

> kernel/nsproxy.c | 17 +++++-

> 7 files changed, 146 insertions(+), 14 deletions(-)

>

> Index: linux-2.6.24/fs/devpts/inode.c

> =====

> --- linux-2.6.24.orig/fs/devpts/inode.c 2008-02-05 19:16:39.000000000 -0800

> +++ linux-2.6.24/fs/devpts/inode.c 2008-02-05 20:27:41.000000000 -0800

> @@ -25,18 +25,25 @@

> #define DEVPTS_SUPER_MAGIC 0x1cd1

>

> extern int pty_limit; /* Config limit on Unix98 ptys */

> -static DEFINE_IDR(allocated_ptys);

> static DECLARE_MUTEX(allocated_ptys_lock);

> +static struct file_system_type devpts_fs_type;

> +

> +struct pts_namespace init_pts_ns = {

> + .kref = {

> + .refcount = ATOMIC_INIT(2),

> + },

> + .allocated_ptys = IDR_INIT(init_pts_ns.allocated_ptys),

```

> + .mnt = NULL,
> +};
>
> static inline struct idr *current_pts_ns_allocated_ptys(void)
> {
> - return &allocated_ptys;
> + return &current->nsproxy->pts_ns->allocated_ptys;
> }
>
> -static struct vfsmount *devpts_mnt;
> static inline struct vfsmount *current_pts_ns_mnt(void)
> {
> - return devpts_mnt;
> + return current->nsproxy->pts_ns->mnt;
> }
>
> static struct {
> @@ -59,6 +66,42 @@ static match_table_t tokens = {
> {Opt_err, NULL}
> };
>
> +struct pts_namespace *new_pts_ns(void)
> +{
> + struct pts_namespace *ns;
> +
> + ns = kmalloc(sizeof(*ns), GFP_KERNEL);
> + if (!ns)
> + return ERR_PTR(-ENOMEM);
> +
> + ns->mnt = kern_mount_data(&devpts_fs_type, ns);

```

You create a circular references here - the namespace holds the vfsmnt, the vfsmnt holds a superblock, a superblock holds the namespace.

```

> + if (IS_ERR(ns->mnt)) {
> + kfree(ns);
> + return ERR_PTR(PTR_ERR(ns->mnt));
> + }
> +
> + idr_init(&ns->allocated_ptys);
> + kref_init(&ns->kref);
> +
> + return ns;
> +}
> +
> +void free_pts_ns(struct kref *ns_kref)
> +{

```

```

> + struct pts_namespace *ns;
> +
> + ns = container_of(ns_kref, struct pts_namespace, kref);
> + BUG_ON(ns == &init_pts_ns);
> +
> + mntput(ns->mnt);
> + /*
> + * TODO:
> + *   idr_remove_all(&ns->allocated_ptys); introduced in 2.6.23
> + */
> + idr_destroy(&ns->allocated_ptys);
> + kfree(ns);
> +}
> +
> static int devpts_remount(struct super_block *sb, int *flags, char *data)
> {
>   char *p;
> @@ -160,18 +203,27 @@ static int devpts_test_sb(struct super_b
>
> static int devpts_set_sb(struct super_block *sb, void *data)
> {
> - sb->s_fs_info = data;
> + struct pts_namespace *ns = data;
> +
> + sb->s_fs_info = get_pts_ns(ns);
>   return set_anon_super(sb, NULL);
> }
>
> static int devpts_get_sb(struct file_system_type *fs_type,
>   int flags, const char *dev_name, void *data, struct vfsmount *mnt)
> {
> + struct pts_namespace *ns;
>   struct super_block *sb;
>   int err;
>
> + /* hereafter we're very similar to proc_get_sb */
> + if (flags & MS_KERNMOUNT)
> +   ns = data;
> + else
> +   ns = current->nsproxy->pts_ns;
> +
>   /* hereafter we're very similar to get_sb_nodev */
> - sb = sget(fs_type, devpts_test_sb, devpts_set_sb, data);
> + sb = sget(fs_type, devpts_test_sb, devpts_set_sb, ns);
>   if (IS_ERR(sb))
>     return PTR_ERR(sb);
>
> @@ -187,16 +239,25 @@ static int devpts_get_sb(struct file_sys

```

```

> }
>
> sb->s_flags |= MS_ACTIVE;
> - devpts_mnt = mnt;
> + ns->mnt = mnt;
>
> return simple_set_mnt(mnt, sb);
> }
>
> +static void devpts_kill_sb(struct super_block *sb)
> +{
> +    struct pts_namespace *ns;
> +
> +    ns = sb->s_fs_info;
> +    kill_anon_super(sb);
> +    put_pts_ns(ns);
> +}
> +
> static struct file_system_type devpts_fs_type = {
> .owner = THIS_MODULE,
> .name = "devpts",
> .get_sb = devpts_get_sb,
> - .kill_sb = kill_anon_super,
> + .kill_sb = devpts_kill_sb,
> };
>
> /*
> @@ -352,18 +413,19 @@ static int __init init_devpts_fs(void)
> if (err)
> return err;
>
> - mnt = kern_mount_data(&devpts_fs_type, NULL);
> + mnt = kern_mount_data(&devpts_fs_type, &init_pts_ns);
> if (IS_ERR(mnt))
> err = PTR_ERR(mnt);
> else
> - devpts_mnt = mnt;
> + init_pts_ns.mnt = mnt;
> return err;
> }
>
> static void __exit exit_devpts_fs(void)
> {
> unregister_filesystem(&devpts_fs_type);
> - mntput(devpts_mnt);
> + mntput(init_pts_ns.mnt);
> + init_pts_ns.mnt = NULL;
> }

```

```

>
> module_init(init_devpts_fs)
> Index: linux-2.6.24/include/linux/devpts_fs.h
> =====
> --- linux-2.6.24.orig/include/linux/devpts_fs.h 2008-02-05 19:16:39.000000000 -0800
> +++ linux-2.6.24/include/linux/devpts_fs.h 2008-02-05 20:21:08.000000000 -0800
> @@ -14,9 +14,45 @@
> #define _LINUX_DEVPTS_FS_H
>
> #include <linux/errno.h>
> +#include <linux/nsproxy.h>
> +#include <linux/kref.h>
> +#include <linux/idr.h>
> +
> +struct pts_namespace {
> + struct kref kref;
> + struct idr allocated_ptys;
> + struct vfsmount *mnt;
> +};
> +
> +extern struct pts_namespace init_pts_ns;
>
> #ifdef CONFIG_UNIX98_PTYS
>
> +extern struct pts_namespace *new_pts_ns(void);
> +extern void free_pts_ns(struct kref *kref);
> +
> +static inline struct pts_namespace *get_pts_ns(struct pts_namespace *ns)
> +{
> + if (ns)
> + kref_get(&ns->kref);
> + return ns;
> +}
> +
> +static inline void put_pts_ns(struct pts_namespace *ns)
> +{
> + if (ns)
> + kref_put(&ns->kref, free_pts_ns);
> +}
> +
> +static inline struct pts_namespace *copy_pts_ns(unsigned long flags,
> + struct pts_namespace *old_ns)
> +{
> + if (flags & CLONE_NEWPTS)
> + return new_pts_ns();
> + else
> + return get_pts_ns(old_ns);
> +}

```

```

> +
> int devpts_new_index(void);
> void devpts_kill_index(int idx);
> int devpts_pty_new(struct tty_struct *tty); /* mknod in devpts */
> @@ -26,6 +62,22 @@ void devpts_pty_kill(int number); /* u
> #else
>
> /* Dummy stubs in the no-pty case */
> +
> +static inline struct pts_namespace *get_pts_ns(struct pts_namespace *ns)
> +{
> + return &init_pts_ns;
> +}
> +
> +static inline void put_pts_ns(struct pts_namespace *ns) { }
> +
> +static inline struct pts_namespace *copy_pts_ns(unsigned long flags,
> + struct pts_namespace *old_ns)
> +{
> + if (flags & CLONE_NEWPTS)
> + return ERR_PTR(-EINVAL);
> + return old_ns;
> +}
> +
> static inline int devpts_new_index(void) { return -EINVAL; }
> static inline void devpts_kill_index(int idx) { }
> static inline int devpts_pty_new(struct tty_struct *tty) { return -EINVAL; }
> Index: linux-2.6.24/include/linux/init_task.h
> =====
> --- linux-2.6.24.orig/include/linux/init_task.h 2008-02-05 19:16:39.000000000 -0800
> +++ linux-2.6.24/include/linux/init_task.h 2008-02-05 19:18:00.000000000 -0800
> @@ -77,6 +77,7 @@ extern struct nsproxy init_nsproxy;
> .mnt_ns = NULL, \
> INIT_NET_NS(net_ns) \
> INIT_IPC_NS(ipc_ns) \
> + .pts_ns = &init_pts_ns, \
> .user_ns = &init_user_ns, \
> }
>
> Index: linux-2.6.24/include/linux/nsproxy.h
> =====
> --- linux-2.6.24.orig/include/linux/nsproxy.h 2008-02-05 19:16:39.000000000 -0800
> +++ linux-2.6.24/include/linux/nsproxy.h 2008-02-05 19:18:00.000000000 -0800
> @@ -8,6 +8,7 @@ struct mnt_namespace;
> struct uts_namespace;
> struct ipc_namespace;
> struct pid_namespace;
> +struct pts_namespace;

```

```

>
> /*
>  * A structure to contain pointers to all per-process
> @@ -29,6 +30,7 @@ struct nsproxy {
>   struct pid_namespace *pid_ns;
>   struct user_namespace *user_ns;
>   struct net      *net_ns;
> + struct pts_namespace *pts_ns;
> };
> extern struct nsproxy init_nsproxy;
>
> Index: linux-2.6.24/include/linux/sched.h
> =====
> --- linux-2.6.24.orig/include/linux/sched.h 2008-02-05 19:16:39.000000000 -0800
> +++ linux-2.6.24/include/linux/sched.h 2008-02-05 19:54:05.000000000 -0800
> @@ -27,6 +27,8 @@
> #define CLONE_NEWUSER 0x10000000 /* New user namespace */
> #define CLONE_NEWPID 0x20000000 /* New pid namespace */
> #define CLONE_NEWNET 0x40000000 /* New network namespace */
> + #define CLONE_NEWPTS (CLONE_NEWNS|0x80000000) /* Temporary - only for patch
review */
> + /* Badly need to /extend clone() !!! */
>
:)

> /*
>  * Scheduling policies
> Index: linux-2.6.24/kernel/fork.c
> =====
> --- linux-2.6.24.orig/kernel/fork.c 2008-02-05 19:16:39.000000000 -0800
> +++ linux-2.6.24/kernel/fork.c 2008-02-05 19:18:00.000000000 -0800
> @@ -1655,7 +1655,7 @@ asmlinkage long sys_unshare(unsigned lon
>   if (unshare_flags & ~(CLONE_THREAD|CLONE_FS|CLONE_NEWNS|CLONE_SIGHAND|
>     CLONE_VM|CLONE_FILES|CLONE_SYSVSEM|
>     CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWUSER|
> -   CLONE_NEWNET))
> +   CLONE_NEWNET|CLONE_NEWPTS))
>   goto bad_unshare_out;
>
>   if ((err = unshare_thread(unshare_flags)))
> Index: linux-2.6.24/kernel/nsproxy.c
> =====
> --- linux-2.6.24.orig/kernel/nsproxy.c 2008-02-05 19:16:39.000000000 -0800
> +++ linux-2.6.24/kernel/nsproxy.c 2008-02-05 19:18:00.000000000 -0800
> @@ -21,6 +21,7 @@
> #include <linux/utsname.h>
> #include <linux/pid_namespace.h>
> #include <net/net_namespace.h>

```

```

> + #include <linux/devpts_fs.h>
>
> static struct kmem_cache *nsproxy_cachep;
>
> @@ -92,8 +93,17 @@ static struct nsproxy *create_new_namesp
> goto out_net;
> }
>
> + new_nsp->pts_ns = copy_pts_ns(flags, tsk->nsproxy->pts_ns);
> + if (IS_ERR(new_nsp->pts_ns)) {
> + err = PTR_ERR(new_nsp->pts_ns);
> + goto out_pts;
> + }
> +
> return new_nsp;
>
> +out_pts:
> + if (new_nsp->net_ns)
> + put_net(new_nsp->net_ns);
> out_net:
> if (new_nsp->user_ns)
> put_user_ns(new_nsp->user_ns);
> @@ -130,7 +140,8 @@ int copy_namespaces(unsigned long flags,
> get_nsproxy(old_ns);
>
> if (!(flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
> - CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET)))
> + CLONE_NEWUSER | CLONE_NEWPID | CLONE_NEWNET |
> + CLONE_NEWPTS)))
> return 0;
>
> if (!capable(CAP_SYS_ADMIN)) {
> @@ -169,6 +180,8 @@ void free_nsproxy(struct nsproxy *ns)
> put_pid_ns(ns->pid_ns);
> if (ns->user_ns)
> put_user_ns(ns->user_ns);
> + if (ns->pts_ns)
> + put_pts_ns(ns->pts_ns);
> put_net(ns->net_ns);
> kmem_cache_free(nsproxy_cachep, ns);
> }
> @@ -183,7 +196,7 @@ int unshare_nsproxy_namespaces(unsigned
> int err = 0;
>
> if (!(unshare_flags & (CLONE_NEWNS | CLONE_NEWUTS | CLONE_NEWIPC |
> - CLONE_NEWUSER | CLONE_NEWNET)))
> + CLONE_NEWUSER | CLONE_NEWNET | CLONE_NEWPTS)))
> return 0;

```

>
> if (!capable(CAP_SYS_ADMIN))
>

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

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