Subject: [patch -mm 0/4] mnt_namespace and pid_namespace
Posted by Cedric Le Goater on Fri, 08 Sep 2006 17:00:04 GMT
View Forum Message <> Reply to Message

all,

here's a cleanup of the pid_namespace and mnt_namespace patches for
the -rc6-mm1.

mnt_namespace is just a global rename. pid_namespace is slowly taking
shape.

thanks,

C.

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

Subject: [patch -mm 1/4] rename struct namespace to struct mnt_namespace
Posted by Cedric Le Goater on Fri, 08 Sep 2006 17:00:05 GMT
View Forum Message <> Reply to Message

this patch renames 'struct namespace' to 'struct mnt_namespace' to
avoid confusion with other namespaces being developped for the
containers : pid, uts, ipc, etc. 'namespace' variables and attributes
are also renamed to 'mnt_ns'

Signed-off-by: Kirill Korotaev <dev@sw.ru>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>

---
 fs/afs/mntpt.c | 2
 fs/cachefiles/cf-bind.c | 2
 fs/cachefiles/cf-proc.c | 3 -
 fs/namespace.c | 109 ++++++++++++++++++++++++++++++----------------
 fs/nfs/getroot.c | 2
 fs/pnode.c | 2
 fs/pnode.h | 2
 fs/proc/base.c | 36 ++++++-------
 fs/reiserfs/super.c | 2
 include/linux/init_task.h | 2
 include/linux/mnt_namespace.h | 42 ++++++++++++++++
 include/linux/mount.h | 4 -
 include/linux/namespace.h | 42 ----------------
 include/linux/nsproxy.h | 4 -

```
 kernel/exit.c               |    2
 kernel/fork.c               |   21 ++++----
 kernel/kmod.c               |    2
 kernel/nsproxy.c            |   16 +++---
 18 files changed, 148 insertions(+), 147 deletions(-)

Index: 2.6.18-rc6-mm1/fs/afs/mntpt.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/afs/mntpt.c
+++ 2.6.18-rc6-mm1/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-mm1/fs/namespace.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/namespace.c
+++ 2.6.18-rc6-mm1/fs/namespace.c
@@ -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>
@@ -133,10 +133,10 @@ 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;
@@ -144,7 +144,7 @@ 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;
@@ -187,19 +187,19 @@ 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)
@@ -320,7 +320,7 @@ 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;

@@ -333,7 +333,7 @@ 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);
@@ -526,8 +526,8 @@ 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--;
@@ -830,7 +830,7 @@ 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);
@@ -1145,9 +1145,9 @@ 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 {
@@ -1168,7 +1168,7 @@ 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)) {
@@ -1178,10 +1178,10 @@ 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);
@@ -1189,7 +1189,7 @@ 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);
  }
 }
@@ -1439,14 +1439,15 @@ 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, *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;

@@ -1457,7 +1458,7 @@ 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);
@@ -1473,10 +1474,10 @@ 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;
@@ -1491,7 +1492,7 @@ 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);
@@ -1506,16 +1507,16 @@ 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;
@@ -1525,16 +1526,16 @@ 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;
 }

@@ -1754,7 +1755,7 @@ 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);
@@ -1779,27 +1780,27 @@ 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)
@@ -1860,11 +1861,11 @@ 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);
@@ -1872,5 +1873,5 @@ 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-mm1/fs/pnode.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/pnode.c
+++ 2.6.18-rc6-mm1/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-mm1/fs/pnode.h
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/pnode.h
+++ 2.6.18-rc6-mm1/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-mm1/fs/proc/base.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/proc/base.c
+++ 2.6.18-rc6-mm1/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>
@@ -364,33 +364,33 @@ 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;
 }
@@ -398,15 +398,15 @@ 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);
@@ -436,20 +436,20 @@ 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->nsproxy->namespace;
-   if (namespace)
-    get_namespace(namespace);
+   mnt_ns = task->nsproxy->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-mm1/fs/reiserfs/super.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/reiserfs/super.c
+++ 2.6.18-rc6-mm1/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-mm1/include/linux/init_task.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/init_task.h
+++ 2.6.18-rc6-mm1/include/linux/init_task.h
@@ -75,7 +75,7 @@ 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)      \
 }

Index: 2.6.18-rc6-mm1/include/linux/mnt_namespace.h
===================================================================
```

```
--- /dev/null
+++ 2.6.18-rc6-mm1/include/linux/mnt_namespace.h
@@ -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
Index: 2.6.18-rc6-mm1/include/linux/mount.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/mount.h
```

```
+++ 2.6.18-rc6-mm1/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-mm1/include/linux/namespace.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/namespace.h
+++ /dev/null
@@ -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
Index: 2.6.18-rc6-mm1/include/linux/nsproxy.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/nsproxy.h
+++ 2.6.18-rc6-mm1/include/linux/nsproxy.h
@@ -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;

Index: 2.6.18-rc6-mm1/kernel/exit.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/exit.c
+++ 2.6.18-rc6-mm1/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>
Index: 2.6.18-rc6-mm1/kernel/fork.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/fork.c
+++ 2.6.18-rc6-mm1/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>
@@ -1507,17 +1507,18 @@ 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;
  }
@@ -1607,7 +1608,7 @@ 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;
@@ -1629,7 +1630,7 @@ 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;
@@ -1670,8 +1671,8 @@ 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;
   }

@@ -1738,7 +1739,7 @@ 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-mm1/kernel/kmod.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/kmod.c
+++ 2.6.18-rc6-mm1/kernel/kmod.c
@@ -25,7 +25,7 @@
 #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-mm1/kernel/nsproxy.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/nsproxy.c
```

```
+++ 2.6.18-rc6-mm1/kernel/nsproxy.c
@@ -17,7 +17,7 @@
 #include <linux/version.h>
 #include <linux/nsproxy.h>
 #include <linux/init_task.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/utsname.h>

 struct nsproxy init_nsproxy = INIT_NSPROXY(init_nsproxy);
@@ -62,8 +62,8 @@ 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)
@@ -99,7 +99,7 @@ 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;

@@ -119,8 +119,8 @@ 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;
  kfree(new_ns);
@@ -129,8 +129,8 @@ 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)
Index: 2.6.18-rc6-mm1/fs/cachefiles/cf-bind.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/cachefiles/cf-bind.c
+++ 2.6.18-rc6-mm1/fs/cachefiles/cf-bind.c
@@ -18,7 +18,7 @@
 #include <linux/file.h>
 #include <linux/namei.h>
 #include <linux/mount.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/statfs.h>
 #include <linux/proc_fs.h>
 #include <linux/ctype.h>
Index: 2.6.18-rc6-mm1/fs/cachefiles/cf-proc.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/cachefiles/cf-proc.c
+++ 2.6.18-rc6-mm1/fs/cachefiles/cf-proc.c
@@ -18,7 +18,7 @@
 #include <linux/file.h>
 #include <linux/namei.h>
 #include <linux/mount.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
 #include <linux/statfs.h>
 #include <linux/proc_fs.h>
 #include <linux/ctype.h>
@@ -70,7 +70,6 @@ static const struct cachefiles_proc_cmd
  { "",  NULL   }
 };

-
 /*****************************************************************************/
 /*
  * do various checks
Index: 2.6.18-rc6-mm1/fs/nfs/getroot.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/nfs/getroot.c
+++ 2.6.18-rc6-mm1/fs/nfs/getroot.c
@@ -31,7 +31,7 @@
 #include <linux/nfs_idmap.h>
 #include <linux/vfs.h>
 #include <linux/namei.h>
-#include <linux/namespace.h>
+#include <linux/mnt_namespace.h>
```

```
#include <linux/security.h>

#include <asm/system.h>

--
```

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

Subject: [patch -mm 2/4] rename struct pspace to struct pid_namespace
Posted by Cedric Le Goater on Fri, 08 Sep 2006 17:00:06 GMT
View Forum Message <> Reply to Message

Rename struct pspace to struct pid_namespace for consistency with
other namespaces (uts_namespace and ipc_namespace). Also rename
include/linux/pspace.h to include/linux/pid_namespace.h and variables
from pspace to pid_ns.

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>

```
 fs/proc/proc_misc.c          |    4 +--
 include/linux/pid_namespace.h |   23 ++++++++++++++++++++
 include/linux/pspace.h       |   23 ----------------------
 kernel/pid.c                 |   49 ++++++++++++++++++++++++++++++++-----------
 4 files changed, 50 insertions(+), 49 deletions(-)

Index: 2.6.18-rc6-mm1/include/linux/pid_namespace.h
===================================================================
--- /dev/null
+++ 2.6.18-rc6-mm1/include/linux/pid_namespace.h
@@ -0,0 +1,23 @@
+#ifndef _LINUX_PID_NS_H
+#define _LINUX_PID_NS_H
+
+#include <linux/sched.h>
+#include <linux/mm.h>
+#include <linux/threads.h>
+#include <linux/pid.h>
+
+struct pidmap {
+       atomic_t nr_free;
+       void *page;
+};
```

```
+
+#define PIDMAP_ENTRIES         ((PID_MAX_LIMIT + 8*PAGE_SIZE - 1)/PAGE_SIZE/8)
+
+struct pid_namespace {
+       struct pidmap pidmap[PIDMAP_ENTRIES];
+       int last_pid;
+};
+
+extern struct pid_namespace init_pid_ns;
+
+#endif /* _LINUX_PID_NS_H */
Index: 2.6.18-rc6-mm1/include/linux/pspace.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/pspace.h
+++ /dev/null
@@ -1,23 +0,0 @@
-#ifndef _LINUX_PSPACE_H
-#define _LINUX_PSPACE_H
-
-#include <linux/sched.h>
-#include <linux/mm.h>
-#include <linux/threads.h>
-#include <linux/pid.h>
-
-struct pidmap {
-       atomic_t nr_free;
-       void *page;
-};
-
-#define PIDMAP_ENTRIES         ((PID_MAX_LIMIT + 8*PAGE_SIZE - 1)/PAGE_SIZE/8)
-
-struct pspace {
-       struct pidmap pidmap[PIDMAP_ENTRIES];
-       int last_pid;
-};
-
-extern struct pspace init_pspace;
-
-#endif /* _LINUX_PSPACE_H */
Index: 2.6.18-rc6-mm1/kernel/pid.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/pid.c
+++ 2.6.18-rc6-mm1/kernel/pid.c
@@ -26,7 +26,7 @@
 #include <linux/init.h>
 #include <linux/bootmem.h>
 #include <linux/hash.h>
-#include <linux/pspace.h>
```

```
+#include <linux/pid_namespace.h>

 #define pid_hashfn(nr) hash_long((unsigned long)nr, pidhash_shift)
 static struct hlist_head *pid_hash;
@@ -43,9 +43,10 @@ int pid_max_max = PID_MAX_LIMIT;
 #define BITS_PER_PAGE  (PAGE_SIZE*8)
 #define BITS_PER_PAGE_MASK (BITS_PER_PAGE-1)

-static inline int mk_pid(struct pspace *pspace, struct pidmap *map, int off)
+static inline int mk_pid(struct pid_namespace *pid_ns,
+  struct pidmap *map, int off)
 {
- return (map - pspace->pidmap)*BITS_PER_PAGE + off;
+ return (map - pid_ns->pidmap)*BITS_PER_PAGE + off;
 }

 #define find_next_offset(map, off)     \
@@ -57,7 +58,7 @@ static inline int mk_pid(struct pspace *
  * value does not cause lots of bitmaps to be allocated, but
  * the scheme scales to up to 4 million PIDs, runtime.
  */
-struct pspace init_pspace = {
+struct pid_namespace init_pid_ns = {
  .pidmap = {
   [ 0 ... PIDMAP_ENTRIES-1] = { ATOMIC_INIT(BITS_PER_PAGE), NULL }
  },
@@ -80,25 +81,25 @@ struct pspace init_pspace = {

 static  __cacheline_aligned_in_smp DEFINE_SPINLOCK(pidmap_lock);

-static fastcall void free_pidmap(struct pspace *pspace, int pid)
+static fastcall void free_pidmap(struct pid_namespace *pid_ns, int pid)
 {
- struct pidmap *map = pspace->pidmap + pid / BITS_PER_PAGE;
+ struct pidmap *map = pid_ns->pidmap + pid / BITS_PER_PAGE;
  int offset = pid & BITS_PER_PAGE_MASK;

  clear_bit(offset, map->page);
  atomic_inc(&map->nr_free);
 }

-static int alloc_pidmap(struct pspace *pspace)
+static int alloc_pidmap(struct pid_namespace *pid_ns)
 {
- int i, offset, max_scan, pid, last = pspace->last_pid;
+ int i, offset, max_scan, pid, last = pid_ns->last_pid;
  struct pidmap *map;
```

```
  pid = last + 1;
  if (pid >= pid_max)
   pid = RESERVED_PIDS;
  offset = pid & BITS_PER_PAGE_MASK;
- map = &pspace->pidmap[pid/BITS_PER_PAGE];
+ map = &pid_ns->pidmap[pid/BITS_PER_PAGE];
  max_scan = (pid_max + BITS_PER_PAGE - 1)/BITS_PER_PAGE - !offset;
  for (i = 0; i <= max_scan; ++i) {
   if (unlikely(!map->page)) {
@@ -120,11 +121,11 @@ static int alloc_pidmap(struct pspace *p
    do {
     if (!test_and_set_bit(offset, map->page)) {
      atomic_dec(&map->nr_free);
-     pspace->last_pid = pid;
+     pid_ns->last_pid = pid;
      return pid;
     }
     offset = find_next_offset(map, offset);
-    pid = mk_pid(pspace, map, offset);
+    pid = mk_pid(pid_ns, map, offset);
    /*
     * find_next_offset() found a bit, the pid from it
     * is in-bounds, and if we fell back to the last
@@ -135,34 +136,34 @@ static int alloc_pidmap(struct pspace *p
      (i != max_scan || pid < last ||
          !((last+1) & BITS_PER_PAGE_MASK)));
   }
-  if (map < &pspace->pidmap[(pid_max-1)/BITS_PER_PAGE]) {
+  if (map < &pid_ns->pidmap[(pid_max-1)/BITS_PER_PAGE]) {
    ++map;
    offset = 0;
   } else {
-   map = &pspace->pidmap[0];
+   map = &pid_ns->pidmap[0];
    offset = RESERVED_PIDS;
    if (unlikely(last == offset))
     break;
   }
-  pid = mk_pid(pspace, map, offset);
+  pid = mk_pid(pid_ns, map, offset);
  }
  return -1;
 }

-static int next_pidmap(struct pspace *pspace, int last)
+static int next_pidmap(struct pid_namespace *pid_ns, int last)
 {
  int offset;
```

```
  struct pidmap *map, *end;

  offset = (last + 1) & BITS_PER_PAGE_MASK;
- map = &pspace->pidmap[(last + 1)/BITS_PER_PAGE];
- end = &pspace->pidmap[PIDMAP_ENTRIES];
+ map = &pid_ns->pidmap[(last + 1)/BITS_PER_PAGE];
+ end = &pid_ns->pidmap[PIDMAP_ENTRIES];
  for (; map < end; map++, offset = 0) {
   if (unlikely(!map->page))
    continue;
   offset = find_next_bit((map)->page, BITS_PER_PAGE, offset);
   if (offset < BITS_PER_PAGE)
-   return mk_pid(pspace, map, offset);
+   return mk_pid(pid_ns, map, offset);
  }
  return -1;
 }
@@ -192,7 +193,7 @@ fastcall void free_pid(struct pid *pid)
  hlist_del_rcu(&pid->pid_chain);
  spin_unlock_irqrestore(&pidmap_lock, flags);

- free_pidmap(&init_pspace, pid->nr);
+ free_pidmap(&init_pid_ns, pid->nr);
  call_rcu(&pid->rcu, delayed_put_pid);
 }

@@ -206,7 +207,7 @@ struct pid *alloc_pid(void)
  if (!pid)
   goto out;

- nr = alloc_pidmap(&init_pspace);
+ nr = alloc_pidmap(&init_pid_ns);
  if (nr < 0)
   goto out_free;

@@ -340,7 +341,7 @@ struct pid *find_ge_pid(int nr)
   pid = find_pid(nr);
   if (pid)
    break;
-  nr = next_pidmap(&init_pspace, nr);
+  nr = next_pidmap(&init_pid_ns, nr);
  } while (nr > 0);

  return pid;
@@ -374,10 +375,10 @@ void __init pidhash_init(void)

 void __init pidmap_init(void)
 {
```

```
- init_pspace.pidmap[0].page = kzalloc(PAGE_SIZE, GFP_KERNEL);
+ init_pid_ns.pidmap[0].page = kzalloc(PAGE_SIZE, GFP_KERNEL);
  /* Reserve PID 0. We never call free_pidmap(0) */
- set_bit(0, init_pspace.pidmap[0].page);
- atomic_dec(&init_pspace.pidmap[0].nr_free);
+ set_bit(0, init_pid_ns.pidmap[0].page);
+ atomic_dec(&init_pid_ns.pidmap[0].nr_free);

  pid_cachep = kmem_cache_create("pid", sizeof(struct pid),
      __alignof__(struct pid),
Index: 2.6.18-rc6-mm1/fs/proc/proc_misc.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/proc/proc_misc.c
+++ 2.6.18-rc6-mm1/fs/proc/proc_misc.c
@@ -45,7 +45,7 @@
 #include <linux/sysrq.h>
 #include <linux/vmalloc.h>
 #include <linux/crash_dump.h>
-#include <linux/pspace.h>
+#include <linux/pid_namespace.h>
 #include <asm/uaccess.h>
 #include <asm/pgtable.h>
 #include <asm/io.h>
@@ -92,7 +92,7 @@ static int loadavg_read_proc(char *page,
   LOAD_INT(a), LOAD_FRAC(a),
   LOAD_INT(b), LOAD_FRAC(b),
   LOAD_INT(c), LOAD_FRAC(c),
-  nr_running(), nr_threads, init_pspace.last_pid);
+  nr_running(), nr_threads, init_pid_ns.last_pid);
  return proc_calc_metrics(page, start, off, count, eof, len);
 }


--
```

Subject: [patch -mm 3/4] add pid_namespace to nsproxy
Posted by Cedric Le Goater on Fri, 08 Sep 2006 17:00:07 GMT
View Forum Message <> Reply to Message

Add a notion of pid namespace to nsproxy (and by extension, to task_struct).
Currently there is only one pid namespace, init_pid_ns and all tasks belong
to this pid namespace. When a new task is created, it inherits its parent's
pid namespace (in copy_process()).

This is based on Eric Biederman's patch: http://lkml.org/lkml/2006/2/6/285

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Cc: Eric Biederman <ebiederm@xmission.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>
Cc: containers@lists.osdl.org

```
 include/linux/init_task.h     |    2 ++
 include/linux/nsproxy.h       |    2 ++
 include/linux/pid_namespace.h |   15 +++++++++++++++
 kernel/nsproxy.c              |   12 ++++++++++++
 4 files changed, 31 insertions(+)

Index: 2.6.18-rc6-mm1/include/linux/init_task.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/init_task.h
+++ 2.6.18-rc6-mm1/include/linux/init_task.h
@@ -7,6 +7,7 @@
 #include <linux/utsname.h>
 #include <linux/lockdep.h>
 #include <linux/ipc.h>
+#include <linux/pid_namespace.h>

 #define INIT_FDTABLE \
 {       \
@@ -72,6 +73,7 @@

 extern struct nsproxy init_nsproxy;
 #define INIT_NSPROXY(nsproxy) {      \
+ .pid_ns  = &init_pid_ns,     \
   .count  = ATOMIC_INIT(1),    \
   .nslock  = SPIN_LOCK_UNLOCKED,    \
   .uts_ns  = &init_uts_ns,     \
Index: 2.6.18-rc6-mm1/include/linux/nsproxy.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/nsproxy.h
+++ 2.6.18-rc6-mm1/include/linux/nsproxy.h
@@ -7,6 +7,7 @@
 struct mnt_namespace;
 struct uts_namespace;
 struct ipc_namespace;
+struct pid_namespace;

 /*
  * A structure to contain pointers to all per-process
```

```
@@ -23,6 +24,7 @@ struct ipc_namespace;
 struct nsproxy {
  atomic_t count;
  spinlock_t nslock;
+ struct pid_namespace *pid_ns;
  struct uts_namespace *uts_ns;
  struct ipc_namespace *ipc_ns;
  struct mnt_namespace *mnt_ns;
Index: 2.6.18-rc6-mm1/include/linux/pid_namespace.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/pid_namespace.h
+++ 2.6.18-rc6-mm1/include/linux/pid_namespace.h
@@ -5,6 +5,7 @@
 #include <linux/mm.h>
 #include <linux/threads.h>
 #include <linux/pid.h>
+#include <linux/nsproxy.h>

 struct pidmap {
        atomic_t nr_free;
@@ -20,4 +21,18 @@ struct pid_namespace {

 extern struct pid_namespace init_pid_ns;

+static inline void get_pid_ns(struct pid_namespace *ns)
+{
+}
+
+static inline int copy_pid_ns(int flags, struct task_struct *tsk)
+{
+ tsk->nsproxy->pid_ns = &init_pid_ns;
+ return 0;
+}
+
+static inline void put_pid_ns(struct pid_namespace *ns)
+{
+}
+
 #endif /* _LINUX_PID_NS_H */
Index: 2.6.18-rc6-mm1/kernel/nsproxy.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/nsproxy.c
+++ 2.6.18-rc6-mm1/kernel/nsproxy.c
@@ -19,6 +19,7 @@
 #include <linux/init_task.h>
 #include <linux/mnt_namespace.h>
 #include <linux/utsname.h>
+#include <linux/pid_namespace.h>
```

```
 struct nsproxy init_nsproxy = INIT_NSPROXY(init_nsproxy);

@@ -68,6 +69,8 @@ struct nsproxy *dup_namespaces(struct ns
    get_uts_ns(ns->uts_ns);
   if (ns->ipc_ns)
    get_ipc_ns(ns->ipc_ns);
+  if (ns->pid_ns)
+   get_pid_ns(ns->pid_ns);
  }

  return ns;
@@ -111,10 +114,17 @@ int copy_namespaces(int flags, struct ta
  if (err)
   goto out_ipc;

+ err = copy_pid_ns(flags, tsk);
+ if (err)
+  goto out_pid;
+
 out:
  put_nsproxy(old_ns);
  return err;

+out_pid:
+ if (new_ns->ipc_ns)
+  put_ipc_ns(new_ns->ipc_ns);
 out_ipc:
  if (new_ns->uts_ns)
   put_uts_ns(new_ns->uts_ns);
@@ -135,5 +145,7 @@ void free_nsproxy(struct nsproxy *ns)
    put_uts_ns(ns->uts_ns);
   if (ns->ipc_ns)
    put_ipc_ns(ns->ipc_ns);
+  if (ns->pid_ns)
+   put_pid_ns(ns->pid_ns);
   kfree(ns);
 }

--
```

Subject: [patch -mm 4/4] add child reaper to pid_namespace

Add per-pid-space child-reaper. This is needed so processes are reaped within
the same pid space and do not spill over to the parent pid space. Its also
needed so containers preserve existing semantic that pid == 1 would reap
orphaned children.

This is based on Eric Biederman's patch: http://lkml.org/lkml/2006/2/6/285

Signed-off-by: Sukadev Bhattiprolu <sukadev@us.ibm.com>
Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Cc: Eric Biederman <ebiederm@xmission.com>
Cc: Dave Hansen <haveblue@us.ibm.com>
Cc: Serge Hallyn <serue@us.ibm.com>

```
 fs/exec.c                  |    5 +++--
 include/linux/pid.h        |    5 +++--
 include/linux/pid_namespace.h |    9 +++++----
 include/linux/sched.h      |    1 -
 init/main.c                |    5 ++---
 kernel/exit.c              |   26 ++++++++++++++++++++--------
 kernel/pid.c               |    3 ++-
 kernel/signal.c            |   10 ++++++++--
 8 files changed, 41 insertions(+), 23 deletions(-)

Index: 2.6.18-rc6-mm1/init/main.c
===================================================================
--- 2.6.18-rc6-mm1.orig/init/main.c
+++ 2.6.18-rc6-mm1/init/main.c
@@ -49,6 +49,7 @@
 #include <linux/buffer_head.h>
 #include <linux/debug_locks.h>
 #include <linux/lockdep.h>
+#include <linux/pid_namespace.h>

 #include <asm/io.h>
 #include <asm/bugs.h>
@@ -623,8 +624,6 @@ static int __init initcall_debug_setup(c
 }
 __setup("initcall_debug", initcall_debug_setup);

-struct task_struct *child_reaper = &init_task;
-
 extern initcall_t __initcall_start[], __initcall_end[];

 static void __init do_initcalls(void)
@@ -724,7 +723,7 @@ static int init(void * unused)
   * assumptions about where in the task array this
```

```
   * can be found.
   */
- child_reaper = current;
+ init_pid_ns.child_reaper = current;

  smp_prepare_cpus(max_cpus);

Index: 2.6.18-rc6-mm1/fs/exec.c
===================================================================
--- 2.6.18-rc6-mm1.orig/fs/exec.c
+++ 2.6.18-rc6-mm1/fs/exec.c
@@ -38,6 +38,7 @@
 #include <linux/binfmts.h>
 #include <linux/swap.h>
 #include <linux/utsname.h>
+#include <linux/pid_namespace.h>
 #include <linux/module.h>
 #include <linux/namei.h>
 #include <linux/proc_fs.h>
@@ -620,8 +621,8 @@ static int de_thread(struct task_struct
   * Reparenting needs write_lock on tasklist_lock,
   * so it is safe to do it under read_lock.
   */
- if (unlikely(tsk->group_leader == child_reaper))
-  child_reaper = tsk;
+ if (unlikely(tsk->group_leader == tsk->nsproxy->pid_ns->child_reaper))
+  tsk->nsproxy->pid_ns->child_reaper = tsk;

  zap_other_threads(tsk);
  read_unlock(&tasklist_lock);
Index: 2.6.18-rc6-mm1/include/linux/pid.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/pid.h
+++ 2.6.18-rc6-mm1/include/linux/pid.h
@@ -35,8 +35,9 @@ enum pid_type
  *
  * Holding a reference to struct pid solves both of these problems.
  * It is small so holding a reference does not consume a lot of
- * resources, and since a new struct pid is allocated when the numeric
- * pid value is reused we don't mistakenly refer to new processes.
+ * resources, and since a new struct pid is allocated when the numeric pid
+ * value is reused (when pids wrap around) we don't mistakenly refer to new
+ * processes.
  */

 struct pid
Index: 2.6.18-rc6-mm1/include/linux/sched.h
===================================================================
```

```
--- 2.6.18-rc6-mm1.orig/include/linux/sched.h
+++ 2.6.18-rc6-mm1/include/linux/sched.h
@@ -1380,7 +1380,6 @@ extern NORET_TYPE void do_group_exit(int
 extern void daemonize(const char *, ...);
 extern int allow_signal(int);
 extern int disallow_signal(int);
-extern struct task_struct *child_reaper;

 extern int do_execve(char *, char __user * __user *, char __user * __user *, struct pt_regs *);
 extern long do_fork(unsigned long, unsigned long, struct pt_regs *, unsigned long, int __user *,
int __user *);
Index: 2.6.18-rc6-mm1/kernel/exit.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/exit.c
+++ 2.6.18-rc6-mm1/kernel/exit.c
@@ -22,6 +22,7 @@
 #include <linux/file.h>
 #include <linux/binfmts.h>
 #include <linux/nsproxy.h>
+#include <linux/pid_namespace.h>
 #include <linux/ptrace.h>
 #include <linux/profile.h>
 #include <linux/mount.h>
@@ -48,7 +49,6 @@
 #include <asm/mmu_context.h>

 extern void sem_exit (void);
-extern struct task_struct *child_reaper;

 static void exit_mm(struct task_struct * tsk);

@@ -259,7 +259,8 @@ static int has_stopped_jobs(int pgrp)
 }

 /**
- * reparent_to_init - Reparent the calling kernel thread to the init task.
+ * reparent_to_init - Reparent the calling kernel thread to the init task
+ * of the pid space that the thread belongs to.
  *
  * If a kernel thread is launched as a result of a system call, or if
  * it ever exits, it should generally reparent itself to init so that
@@ -277,8 +278,8 @@ static void reparent_to_init(void)
  ptrace_unlink(current);
  /* Reparent to init */
  remove_parent(current);
- current->parent = child_reaper;
- current->real_parent = child_reaper;
+ current->parent = current->nsproxy->pid_ns->child_reaper;
```

```
+ current->real_parent = current->nsproxy->pid_ns->child_reaper;
  add_parent(current);

  /* Set the exit signal to SIGCHLD so we signal init on exit */
@@ -662,7 +663,8 @@ reparent_thread(struct task_struct *p, s
  * When we die, we re-parent all our children.
  * Try to give them to another thread in our thread
  * group, and if no such member exists, give it to
- * the global child reaper process (ie "init")
+ * the child reaper process (ie "init") in our pid
+ * space.
  */
 static void
 forget_original_parent(struct task_struct *father, struct list_head *to_release)
@@ -673,7 +675,10 @@ forget_original_parent(struct task_struc
  do {
   reaper = next_thread(reaper);
   if (reaper == father) {
-   reaper = child_reaper;
+   /*
+    * FIXME: which reaper to use ?
+    */
+   reaper = init_pid_ns.child_reaper;
    break;
   }
  } while (reaper->exit_state);
@@ -861,8 +866,13 @@ fastcall NORET_TYPE void do_exit(long co
   panic("Aiee, killing interrupt handler!");
  if (unlikely(!tsk->pid))
   panic("Attempted to kill the idle task!");
- if (unlikely(tsk == child_reaper))
-  panic("Attempted to kill init!");
+ if (unlikely(tsk == tsk->nsproxy->pid_ns->child_reaper)) {
+  if (tsk->nsproxy->pid_ns != &init_pid_ns)
+   tsk->nsproxy->pid_ns->child_reaper = init_pid_ns.child_reaper;
+  else
+   panic("Attempted to kill init!");
+ }
+

  if (unlikely(current->ptrace & PT_TRACE_EXIT)) {
   current->ptrace_message = code;
Index: 2.6.18-rc6-mm1/kernel/signal.c
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/signal.c
+++ 2.6.18-rc6-mm1/kernel/signal.c
@@ -24,6 +24,8 @@
 #include <linux/ptrace.h>
```

```
 #include <linux/signal.h>
 #include <linux/capability.h>
+#include <linux/pid_namespace.h>
+#include <linux/nsproxy.h>
 #include <asm/param.h>
 #include <asm/uaccess.h>
 #include <asm/unistd.h>
@@ -1994,8 +1996,12 @@ relock:
   if (sig_kernel_ignore(signr)) /* Default is nothing. */
    continue;

-  /* Init gets no signals it doesn't want.  */
-  if (current == child_reaper)
+  /*
+   * Init of a pid space gets no signals it doesn't want from
+   * within that pid space. It can of course get signals from
+   * its parent pid space.
+   */
+  if (current == current->nsproxy->pid_ns->child_reaper)
    continue;

   if (sig_kernel_stop(signr)) {
Index: 2.6.18-rc6-mm1/include/linux/pid_namespace.h
===================================================================
--- 2.6.18-rc6-mm1.orig/include/linux/pid_namespace.h
+++ 2.6.18-rc6-mm1/include/linux/pid_namespace.h
@@ -8,15 +8,16 @@
 #include <linux/nsproxy.h>

 struct pidmap {
-       atomic_t nr_free;
-       void *page;
+ atomic_t nr_free;
+ void *page;
 };

 #define PIDMAP_ENTRIES         ((PID_MAX_LIMIT + 8*PAGE_SIZE - 1)/PAGE_SIZE/8)

 struct pid_namespace {
-       struct pidmap pidmap[PIDMAP_ENTRIES];
-       int last_pid;
+ struct pidmap pidmap[PIDMAP_ENTRIES];
+ int last_pid;
+ struct task_struct * child_reaper;
 };

 extern struct pid_namespace init_pid_ns;
Index: 2.6.18-rc6-mm1/kernel/pid.c
```

```
===================================================================
--- 2.6.18-rc6-mm1.orig/kernel/pid.c
+++ 2.6.18-rc6-mm1/kernel/pid.c
@@ -62,7 +62,8 @@ struct pid_namespace init_pid_ns = {
  .pidmap = {
   [ 0 ... PIDMAP_ENTRIES-1] = { ATOMIC_INIT(BITS_PER_PAGE), NULL }
  },
- .last_pid = 0
+ .last_pid = 0,
+ .child_reaper = &init_task
 };

 /*

--
```

_____

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