Subject: [PATCH 10/15] Make each namespace has its own proc tree
Posted by Pavel Emelianov on Thu, 26 Jul 2007 14:54:22 GMT
View Forum Message <> Reply to Message

The idea is the following: when a namespace is created the
new proc superblock, pointing to this namespace, is created.
When user accesses the proc tree from some pountmoint it oberves
the tasks that live in the namespace, pointer by the according
spuperblock.

When task dies, its pids must be flushed from several proc trees,
i.e. the ones that are owned by the namespaces that can see the
taks's struct pid.

Having the namespaces live without the proc mount is racy, so
during the namespace creation we kern_mount the new proc instance.

We cannot just get the reference on the struct pid_namespace from
the superblock, otherwise we'll have the ns->proc_mnt->sb->ns
loop. To break this, when the init task dies, it releases the
kern_mount-ed instance.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
 fs/proc/base.c              |   12 ++++++
 fs/proc/inode.c             |    3 +
 fs/proc/root.c              |   83 ++++++++++++++++++++++++++++++++++++++++++---
 include/linux/pid_namespace.h |    3 +
 include/linux/proc_fs.h     |   15 +++++++
 kernel/exit.c               |   18 ++++++++-
 kernel/fork.c               |    5 ++
 kernel/pid.c                |    1
 8 files changed, 133 insertions(+), 7 deletions(-)

diff -upr linux-2.6.23-rc1-mm1.orig/fs/proc/base.c linux-2.6.23-rc1-mm1-7/fs/proc/base.c
--- linux-2.6.23-rc1-mm1.orig/fs/proc/base.c 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/fs/proc/base.c 2007-07-26 16:36:37.000000000 +0400
@@ -2161,7 +2162,19 @@ out:

 void proc_flush_task(struct task_struct *task, struct pid *pid)
 {
+ int i;
+ struct upid *upid;
+
  proc_flush_task_mnt(task, proc_mnt);
+ if (pid == NULL)
```

```
+  return;
+
+ for (i = 1; i <= pid->level; i++)
+  proc_flush_task_mnt(task, pid->numbers[i].ns->proc_mnt);
+
+ upid = &pid->numbers[pid->level];
+ if (upid->nr == 1)
+  pid_ns_release_proc(upid->ns);
 }

 static struct dentry *proc_pid_instantiate(struct inode *dir,
diff -upr linux-2.6.23-rc1-mm1.orig/fs/proc/inode.c linux-2.6.23-rc1-mm1-7/fs/proc/inode.c
--- linux-2.6.23-rc1-mm1.orig/fs/proc/inode.c 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/fs/proc/inode.c 2007-07-26 16:36:36.000000000 +0400
@@ -16,6 +16,7 @@
 #include <linux/init.h>
 #include <linux/module.h>
 #include <linux/smp_lock.h>
+#include <linux/pid_namespace.h>

 #include <asm/system.h>
 #include <asm/uaccess.h>
@@ -427,7 +428,7 @@ out_mod:
  return NULL;
 }

-int proc_fill_super(struct super_block *s, void *data, int silent)
+int proc_fill_super(struct super_block *s)
 {
  struct inode * root_inode;

diff -upr linux-2.6.23-rc1-mm1.orig/fs/proc/root.c linux-2.6.23-rc1-mm1-7/fs/proc/root.c
--- linux-2.6.23-rc1-mm1.orig/fs/proc/root.c 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/fs/proc/root.c 2007-07-26 16:36:37.000000000 +0400
@@ -18,32 +18,90 @@
 #include <linux/bitops.h>
 #include <linux/smp_lock.h>
 #include <linux/mount.h>
+#include <linux/pid_namespace.h>

 #include "internal.h"

 struct proc_dir_entry *proc_net, *proc_net_stat, *proc_bus, *proc_root_fs, *proc_root_driver;

+static int proc_test_super(struct super_block *sb, void *data)
+{
+ return sb->s_fs_info == data;
+}
```

```
+
+static int proc_set_super(struct super_block *sb, void *data)
+{
+ struct pid_namespace *ns;
+
+ ns = (struct pid_namespace *)data;
+ sb->s_fs_info = get_pid_ns(ns);
+ return set_anon_super(sb, NULL);
+}
+
 static int proc_get_sb(struct file_system_type *fs_type,
  int flags, const char *dev_name, void *data, struct vfsmount *mnt)
 {
+ int err;
+ struct super_block *sb;
+ struct pid_namespace *ns;
+ struct proc_inode *ei;
+
  if (proc_mnt) {
   /* Seed the root directory with a pid so it doesn't need
    * to be special in base.c.  I would do this earlier but
    * the only task alive when /proc is mounted the first time
    * is the init_task and it doesn't have any pids.
    */
-  struct proc_inode *ei;
   ei = PROC_I(proc_mnt->mnt_sb->s_root->d_inode);
   if (!ei->pid)
    ei->pid = find_get_pid(1);
  }
- return get_sb_single(fs_type, flags, data, proc_fill_super, mnt);
+
+ if (flags & MS_KERNMOUNT)
+  ns = (struct pid_namespace *)data;
+ else
+  ns = current->nsproxy->pid_ns;
+
+ sb = sget(fs_type, proc_test_super, proc_set_super, ns);
+ if (IS_ERR(sb))
+  return PTR_ERR(sb);
+
+ if (!sb->s_root) {
+  sb->s_flags = flags;
+  err = proc_fill_super(sb);
+  if (err) {
+   up_write(&sb->s_umount);
+   deactivate_super(sb);
+   return err;
+  }
```

```
+
+  ei = PROC_I(sb->s_root->d_inode);
+  if (!ei->pid) {
+   rcu_read_lock();
+   ei->pid = get_pid(find_pid_ns(1, ns));
+   rcu_read_unlock();
+  }
+
+  sb->s_flags |= MS_ACTIVE;
+  ns->proc_mnt = mnt;
+ }
+
+ return simple_set_mnt(mnt, sb);
+}
+
+static void proc_kill_sb(struct super_block *sb)
+{
+ struct pid_namespace *ns;
+
+ ns = (struct pid_namespace *)sb->s_fs_info;
+ kill_anon_super(sb);
+ put_pid_ns(ns);
 }

 static struct file_system_type proc_fs_type = {
  .name  = "proc",
  .get_sb  = proc_get_sb,
- .kill_sb = kill_anon_super,
+ .kill_sb = proc_kill_sb,
 };

 void __init proc_root_init(void)
@@ -54,12 +112,13 @@ void __init proc_root_init(void)
  err = register_filesystem(&proc_fs_type);
  if (err)
   return;
- proc_mnt = kern_mount(&proc_fs_type);
+ proc_mnt = kern_mount_data(&proc_fs_type, &init_pid_ns);
  err = PTR_ERR(proc_mnt);
  if (IS_ERR(proc_mnt)) {
   unregister_filesystem(&proc_fs_type);
   return;
  }
+
  proc_misc_init();
  proc_net = proc_mkdir("net", NULL);
  proc_net_stat = proc_mkdir("net/stat", NULL);
@@ -153,6 +212,22 @@ struct proc_dir_entry proc_root = {
```

```
  .parent  = &proc_root,
 };

+int pid_ns_prepare_proc(struct pid_namespace *ns)
+{
+ struct vfsmount *mnt;
+
+ mnt = kern_mount_data(&proc_fs_type, ns);
+ if (IS_ERR(mnt))
+  return PTR_ERR(mnt);
+
+ return 0;
+}
+
+void pid_ns_release_proc(struct pid_namespace *ns)
+{
+ mntput(ns->proc_mnt);
+}
+
 EXPORT_SYMBOL(proc_symlink);
 EXPORT_SYMBOL(proc_mkdir);
 EXPORT_SYMBOL(create_proc_entry);
diff -upr linux-2.6.23-rc1-mm1.orig/include/linux/pid_namespace.h
linux-2.6.23-rc1-mm1-7/include/linux/pid_namespace.h
--- linux-2.6.23-rc1-mm1.orig/include/linux/pid_namespace.h 2007-07-26 16:34:45.000000000
+0400
+++ linux-2.6.23-rc1-mm1-7/include/linux/pid_namespace.h 2007-07-26 16:36:36.000000000
+0400
@@ -16,6 +15,9 @@ struct pidmap {
  int last_pid;
  struct task_struct *child_reaper;
  struct kmem_cache *pid_cachep;
+#ifdef CONFIG_PROC_FS
+ struct vfsmount *proc_mnt;
+#endif
 };

 extern struct pid_namespace init_pid_ns;
diff -upr linux-2.6.23-rc1-mm1.orig/include/linux/proc_fs.h
linux-2.6.23-rc1-mm1-7/include/linux/proc_fs.h
--- linux-2.6.23-rc1-mm1.orig/include/linux/proc_fs.h 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/include/linux/proc_fs.h 2007-07-26 16:36:36.000000000 +0400
@@ -126,7 +126,8 @@ extern struct proc_dir_entry *create_pro
 extern void remove_proc_entry(const char *name, struct proc_dir_entry *parent);

 extern struct vfsmount *proc_mnt;
-extern int proc_fill_super(struct super_block *,void *,int);
+struct pid_namespace;
```

```
+extern int proc_fill_super(struct super_block *);
 extern struct inode *proc_get_inode(struct super_block *, unsigned int, struct proc_dir_entry *);

 /*
@@ -143,6 +144,9 @@ extern const struct file_operations proc
 extern const struct file_operations proc_kmsg_operations;
 extern const struct file_operations ppc_htab_operations;

+extern int pid_ns_prepare_proc(struct pid_namespace *ns);
+extern void pid_ns_release_proc(struct pid_namespace *ns);
+
 /*
  * proc_tty.c
  */
@@ -248,6 +254,15 @@ static inline void proc_tty_unregister_d

 extern struct proc_dir_entry proc_root;

+static inline int pid_ns_prepare_proc(struct pid_namespace *ns)
+{
+ return 0;
+}
+
+static inline void pid_ns_release_proc(struct pid_namespace *ns)
+{
+}
+
 #endif /* CONFIG_PROC_FS */

 #if !defined(CONFIG_PROC_KCORE)
diff -upr linux-2.6.23-rc1-mm1.orig/kernel/exit.c linux-2.6.23-rc1-mm1-7/kernel/exit.c
--- linux-2.6.23-rc1-mm1.orig/kernel/exit.c 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/kernel/exit.c 2007-07-26 16:36:37.000000000 +0400
@@ -153,6 +153,7 @@ static void delayed_put_task_struct(stru

 void release_task(struct task_struct * p)
 {
+ struct pid *pid;
  struct task_struct *leader;
  int zap_leader;
 repeat:
@@ -160,6 +161,20 @@ repeat:
  write_lock_irq(&tasklist_lock);
  ptrace_unlink(p);
  BUG_ON(!list_empty(&p->ptrace_list) || !list_empty(&p->ptrace_children));
+ /*
+  * we have to keep this pid till proc_flush_task() to make
+  * it possible to flush all dentries holding it. pid will
```

```
+  * be put ibidem
+  *
+  * however if the pid belogs to init namespace only, we can
+  * optimize this out
+  */
+ pid = task_pid(p);
+ if (pid->level != 0)
+  get_pid(pid);
+ else
+  pid = NULL;
+
   __exit_signal(p);

   /*
@@ -184,7 +199,8 @@ repeat:
   }

   write_unlock_irq(&tasklist_lock);
- proc_flush_task(p, NULL);
+ proc_flush_task(p, pid);
+ put_pid(pid);
   release_thread(p);
   call_rcu(&p->rcu, delayed_put_task_struct);

diff -upr linux-2.6.23-rc1-mm1.orig/kernel/fork.c linux-2.6.23-rc1-mm1-7/kernel/fork.c
--- linux-2.6.23-rc1-mm1.orig/kernel/fork.c 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/kernel/fork.c 2007-07-26 16:36:37.000000000 +0400
@@ -50,6 +50,7 @@
 #include <linux/taskstats_kern.h>
 #include <linux/random.h>
 #include <linux/tty.h>
+#include <linux/proc_fs.h>

 #include <asm/pgtable.h>
 #include <asm/pgalloc.h>
@@ -1141,6 +1131,10 @@ static struct task_struct *copy_process(
    pid = alloc_pid(task_active_pid_ns(p));
    if (!pid)
     goto bad_fork_cleanup_namespaces;
+
+  if (clone_flags & CLONE_NEWPID)
+   if (pid_ns_prepare_proc(task_active_pid_ns(p)))
+    goto bad_fork_free_pid;
   }

   p->pid = pid_nr(pid);
diff -upr linux-2.6.23-rc1-mm1.orig/kernel/pid.c linux-2.6.23-rc1-mm1-7/kernel/pid.c
--- linux-2.6.23-rc1-mm1.orig/kernel/pid.c 2007-07-26 16:34:45.000000000 +0400
```

```
+++ linux-2.6.23-rc1-mm1-7/kernel/pid.c 2007-07-26 16:36:37.000000000 +0400
@@ -28,6 +28,7 @@
 #include <linux/hash.h>
 #include <linux/pid_namespace.h>
 #include <linux/init_task.h>
+#include <linux/proc_fs.h>

 #define pid_hashfn(nr, ns) \
  hash_long((unsigned long)nr + (unsigned long)ns, pidhash_shift)
```