Subject: [PATCH 12/20] Make proc hav emultiple superblocks - one for each namespace
Posted by Pavel Emelianov on Tue, 07 Aug 2007 09:29:58 GMT
View Forum Message <> Reply to Message

Each pid namespace have to be visible through its own proc mount.
Thus we need to have per-namespace proc trees with their own
superblocks.

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

```
 fs/proc/inode.c              |    2 -
 fs/proc/root.c               |   67 ++++++++++++++++++++++++++++++++++++++++---
 include/linux/pid_namespace.h |    3 +
 include/linux/proc_fs.h      |    3 +
 4 files changed, 69 insertions(+), 6 deletions(-)

--- ./fs/proc/inode.c.ve12 2007-08-06 14:54:42.000000000 +0400
+++ ./fs/proc/inode.c 2007-08-06 14:58:51.000000000 +0400
@@ -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;

--- ./fs/proc/root.c.ve12 2007-08-06 14:54:42.000000000 +0400
+++ ./fs/proc/root.c 2007-08-06 14:58:51.000000000 +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);
--- ./include/linux/pid_namespace.h.ve12 2007-08-06 14:58:51.000000000 +0400
+++ ./include/linux/pid_namespace.h 2007-08-06 14:59:02.000000000 +0400
@@ -23,6 +23,9 @@ struct pid_namespace {
```

```
  struct kmem_cache *pid_cachep;
  int level;
  struct pid_namespace *parent;
+#ifdef CONFIG_PROC_FS
+ struct vfsmount *proc_mnt;
+#endif
 };

 extern struct pid_namespace init_pid_ns;
--- ./include/linux/proc_fs.h.ve12 2007-08-06 14:58:51.000000000 +0400
+++ ./include/linux/proc_fs.h 2007-08-06 14:58:51.000000000 +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 *);

 /*
```