Subject: [RFC] v2 - [PATCH 3/3] introduce proc_mnt for pid_ns
Posted by Dave Hansen on Thu, 01 Feb 2007 02:53:17 GMT
View Forum Message <> Reply to Message

The following patch completes the removal of the global proc_mnt.
It fetches the mnt on which to do dentry invalidations from the
pid_namespace in which the task appears.

For now, there is only one pid namespace in mainline so this is
straightforward. In the -lxc tree we'll have to do something
more complex.

Note that the new proc_compare_super() enforces the "one proc sb
per pid_namespace" limit.

/proc currently has some special code to make sure that the root
directory gets set up correctly. It proc_mnt variable in order
to find its way to the root inode.

Now that we don't have the global proc_mnt, we can fill in the
root inode's data in proc_fill_super(), where it takes a wee bit
less work than in proc_get_sb().

```
 lxc-dave/fs/proc/base.c              |   26 ++++++++++++++++++--
 lxc-dave/fs/proc/inode.c             |   11 ++++++--
 lxc-dave/fs/proc/root.c              |   45 +++++++++++++++++++----------------
 lxc-dave/include/linux/pid_namespace.h |    1
 lxc-dave/include/linux/proc_fs.h     |    1
 5 files changed, 61 insertions(+), 23 deletions(-)

diff -puN fs/proc/base.c~A3-remove-proc_mnt-1 fs/proc/base.c
--- lxc/fs/proc/base.c~A3-remove-proc_mnt-1 2007-01-31 17:29:43.000000000 -0800
+++ lxc-dave/fs/proc/base.c 2007-01-31 17:29:43.000000000 -0800
@@ -70,6 +70,7 @@
 #include <linux/seccomp.h>
 #include <linux/cpuset.h>
 #include <linux/audit.h>
+#include <linux/pid_namespace.h>
 #include <linux/poll.h>
 #include <linux/nsproxy.h>
 #include <linux/oom.h>
@@ -1905,9 +1906,11 @@ static struct inode_operations proc_tgid
 };

 /**
```

```
- * proc_flush_task -  Remove dcache entries for @task from the /proc dcache.
+ * proc_flush_task_from_pid_ns -  Remove dcache entries for @task
+ *       from the /proc dcache.
  *
  * @task: task that should be flushed.
+ * @pid_ns: pid_namespace in which that task appears
  *
  * Looks in the dcache for
  * /proc/@pid
@@ -1925,11 +1928,20 @@ static struct inode_operations proc_tgid
  *       that no dcache entries will exist at process exit time it
  *       just makes it very unlikely that any will persist.
  */
-void proc_flush_task(struct task_struct *task)
+void proc_flush_task_from_pid_ns(struct task_struct *task,
+    struct pid_namespace* pid_ns)
 {
  struct dentry *dentry, *leader, *dir;
  char buf[PROC_NUMBUF];
  struct qstr name;
+ struct vfsmount *proc_mnt = pid_ns->proc_mnt;
+
+ /*
+  * It is possible that no /procs have been instantiated
+  * for this particular pid namespace.
+  */
+ if (!proc_mnt)
+  return;

  name.name = buf;
  name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
@@ -1971,6 +1983,16 @@ out:
  return;
 }

+void proc_flush_task(struct task_struct *task)
+{
+ /*
+  * Right now, tasks only appear in their own pid_ns.
+  * With containers this function will change to a loop
+  * over all pid_ns's in which the task appears.
+  */
+ proc_flush_task_from_pid_ns(task, current->nsproxy->pid_ns);
+}
+
 static struct dentry *proc_pid_instantiate(struct inode *dir,
        struct dentry * dentry,
        struct task_struct *task, void *ptr)
```

```
diff -puN fs/proc/inode.c~A3-remove-proc_mnt-1 fs/proc/inode.c
--- lxc/fs/proc/inode.c~A3-remove-proc_mnt-1 2007-01-31 17:29:43.000000000 -0800
+++ lxc-dave/fs/proc/inode.c 2007-01-31 17:34:56.000000000 -0800
@@ -67,8 +67,6 @@ static void proc_delete_inode(struct ino
  clear_inode(inode);
 }

-struct vfsmount *proc_mnt;
-
 static void proc_read_inode(struct inode * inode)
 {
  inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;
@@ -183,6 +181,7 @@ out_mod:

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

  s->s_flags |= MS_NODIRATIME | MS_NOSUID | MS_NOEXEC;
@@ -200,6 +199,14 @@ int proc_fill_super(struct super_block *
  s->s_root = d_alloc_root(root_inode);
  if (!s->s_root)
   goto out_no_root;
+ /* 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.
+  */
+ ei = PROC_I(root_inode);
+ if (!ei->pid)
+  ei->pid = find_get_pid(1);
  return 0;

 out_no_root:
diff -puN fs/proc/root.c~A3-remove-proc_mnt-1 fs/proc/root.c
--- lxc/fs/proc/root.c~A3-remove-proc_mnt-1 2007-01-31 17:29:43.000000000 -0800
+++ lxc-dave/fs/proc/root.c 2007-01-31 17:34:21.000000000 -0800
@@ -18,6 +18,7 @@
 #include <linux/bitops.h>
 #include <linux/smp_lock.h>
 #include <linux/mount.h>
+#include <linux/pid_namespace.h>

 #include "internal.h"

@@ -27,21 +28,35 @@ struct proc_dir_entry *proc_net, *proc_n
 struct proc_dir_entry *proc_sys_root;
```

```
 #endif

+static int proc_compare_super(struct super_block *s, void *p)
+{
+ struct pid_namespace *pid_ns = p;
+ if (pid_ns->proc_mnt->mnt_sb == s)
+  return 1;
+ return 0;
+}
+
 static int proc_get_sb(struct file_system_type *fs_type,
  int flags, const char *dev_name, void *data, struct vfsmount *mnt)
 {
- 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);
+ struct super_block *s;
+ struct pid_namespace *pid_ns = current->nsproxy->pid_ns;
+ int error;
+
+ s = sget(fs_type, proc_compare_super, set_anon_super, pid_ns);
+ if (IS_ERR(s))
+  return PTR_ERR(s);
+ if (!pid_ns->proc_mnt)
+  pid_ns->proc_mnt = mnt;
+
+ error = fill_if_new_sb(s, pid_ns, flags, proc_fill_super);
+ if (error)
+  return error;
+
+ do_remount_sb(s, flags, data, 0);
+ error = simple_set_mnt(mnt, s);
+
+ return error;
 }

 static struct file_system_type proc_fs_type = {
@@ -58,12 +73,6 @@ void __init proc_root_init(void)
  err = register_filesystem(&proc_fs_type);
```

```
  if (err)
   return;
- proc_mnt = kern_mount(&proc_fs_type);
- 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);
diff -puN include/linux/pid_namespace.h~A3-remove-proc_mnt-1 include/linux/pid_namespace.h
--- lxc/include/linux/pid_namespace.h~A3-remove-proc_mnt-1 2007-01-31 17:29:43.000000000
-0800
+++ lxc-dave/include/linux/pid_namespace.h 2007-01-31 17:29:43.000000000 -0800
@@ -20,6 +20,7 @@ struct pid_namespace {
  struct pidmap pidmap[PIDMAP_ENTRIES];
  int last_pid;
  struct task_struct *child_reaper;
+ struct vfsmount *proc_mnt;
 };

 extern struct pid_namespace init_pid_ns;
diff -puN include/linux/proc_fs.h~A3-remove-proc_mnt-1 include/linux/proc_fs.h
--- lxc/include/linux/proc_fs.h~A3-remove-proc_mnt-1 2007-01-31 17:29:43.000000000 -0800
+++ lxc-dave/include/linux/proc_fs.h 2007-01-31 17:32:22.000000000 -0800
@@ -109,7 +109,6 @@ extern struct proc_dir_entry *create_pro
       struct proc_dir_entry *parent);
 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);
 extern struct inode *proc_get_inode(struct super_block *, unsigned int, struct proc_dir_entry *);

diff -puN include/linux/fs.h~A3-remove-proc_mnt-1 include/linux/fs.h
diff -puN include/linux/mount.h~A3-remove-proc_mnt-1 include/linux/mount.h
_
```