Subject: [RFC][PATCH] Do not set /proc inode->pid for non-pid-related inodes
Posted by Dave Hansen on Mon, 19 Mar 2007 22:27:47 GMT
View Forum Message <> Reply to Message

I was tracking down why we need find_get_pid(1) in
proc_get_sb(), when I realized that we apparently
don't need a pid at all in the non-pid parts of /proc.

Anyone see any problems with this approach?

For what I would imagine are historical reasons, we set
all struct proc_inode->pid fields. We use the init
process for all non-/proc/<pid> inodes.

We get a handle to the init process in proc_get_sb()
then fetch it out in proc_pid_readdir():

```
 struct task_struct *reaper = get_proc_task(filp->f_path.dentry->d_inode);
```

The filp in that case is always the root inode on which
someone is doing a readdir. This reaper variable gets
passed down into proc_base_instantiate() and eventually
set in the new inode's ->pid field.

The problem is that I don't see anywhere that we
actually go and use this, outside of the /proc/<pid>
directories. Just referencing the init process like
this is a pain for containers because our init process
(pid == 1) can actually go away.

So, this patch removes all non-pid-dir use of
proc_inode->pid. It puts a WARN_ON() in case anyone
tries to instantiate a proc inode with a pid in a place
we don't expect there to be one.

```
 lxc-dave/fs//proc/inode.c |    6 ------
 lxc-dave/fs/proc/base.c   |   31 ++++++++++----------------------
 2 files changed, 10 insertions(+), 27 deletions(-)

diff -puN fs//proc/inode.c~funny-proc-patch fs//proc/inode.c
--- lxc/fs//proc/inode.c~funny-proc-patch 2007-03-19 15:10:50.000000000 -0700
+++ lxc-dave/fs//proc/inode.c 2007-03-19 15:10:50.000000000 -0700
@@ -184,7 +184,6 @@ out_mod:
 int proc_fill_super(struct super_block *s, void *data, int silent)
```

```
 {
  struct pid_namespace *pid_ns = data;
- struct proc_inode *ei;
  struct inode * root_inode;

  s->s_flags |= MS_NODIRATIME | MS_NOSUID | MS_NOEXEC;
@@ -204,11 +203,6 @@ 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.
-  */
- ei = PROC_I(root_inode);
- ei->pid = find_get_pid(1);
  return 0;

 out_no_root:
diff -puN fs//proc/internal.h~funny-proc-patch fs//proc/internal.h
diff -puN fs/proc/base.c~funny-proc-patch fs/proc/base.c
--- lxc/fs/proc/base.c~funny-proc-patch 2007-03-19 15:10:50.000000000 -0700
+++ lxc-dave/fs/proc/base.c 2007-03-19 15:11:40.000000000 -0700
@@ -1171,11 +1171,15 @@ static int pid_revalidate(struct dentry

 static int pid_delete_dentry(struct dentry * dentry)
 {
+ struct pid *pid;
  /* Is the task we represent dead?
   * If so, then don't put the dentry on the lru list,
   * kill it immediately.
   */
- return !proc_pid(dentry->d_inode)->tasks[PIDTYPE_PID].first;
+ pid = proc_pid(dentry->d_inode);
+ if (!pid)
+  return 0;
+ return !pid->tasks[PIDTYPE_PID].first;
 }

 static struct dentry_operations pid_dentry_operations =
@@ -1813,6 +1817,7 @@ static struct dentry *proc_base_instanti
  struct proc_inode *ei;
  struct dentry *error = ERR_PTR(-EINVAL);

+ WARN_ON(task);
  /* Allocate the inode */
  error = ERR_PTR(-ENOMEM);
  inode = new_inode(dir->i_sb);
@@ -1823,13 +1828,6 @@ static struct dentry *proc_base_instanti
```

```
  ei = PROC_I(inode);
  inode->i_mtime = inode->i_atime = inode->i_ctime = CURRENT_TIME;

- /*
-  * grab the reference to the task.
-  */
- ei->pid = get_task_pid(task, PIDTYPE_PID);
- if (!ei->pid)
-  goto out_iput;
-
  inode->i_uid = 0;
  inode->i_gid = 0;
  inode->i_mode = p->mode;
@@ -1847,9 +1845,6 @@ static struct dentry *proc_base_instanti
  error = NULL;
 out:
  return error;
-out_iput:
- iput(inode);
- goto out;
 }

 static struct dentry *proc_base_lookup(struct inode *dir, struct dentry *dentry)
@@ -1874,7 +1869,7 @@ static struct dentry *proc_base_lookup(s
  if (p > last)
   goto out;

- error = proc_base_instantiate(dir, dentry, task, p);
+ error = proc_base_instantiate(dir, dentry, NULL, p);

 out:
  put_task_struct(task);
@@ -1883,10 +1878,10 @@ out_no_task:
 }

 static int proc_base_fill_cache(struct file *filp, void *dirent, filldir_t filldir,
- struct task_struct *task, struct pid_entry *p)
+ struct pid_entry *p)
 {
  return proc_fill_cache(filp, dirent, filldir, p->name, p->len,
-    proc_base_instantiate, task, p);
+    proc_base_instantiate, NULL, p);
 }

 #ifdef CONFIG_TASK_IO_ACCOUNTING
@@ -2197,16 +2192,12 @@ static int proc_pid_fill_cache(struct fi
 int proc_pid_readdir(struct file * filp, void * dirent, filldir_t filldir)
 {
```

```
  unsigned int nr = filp->f_pos - FIRST_PROCESS_ENTRY;
- struct task_struct *reaper = get_proc_task(filp->f_path.dentry->d_inode);
  struct task_struct *task;
  int tgid;

- if (!reaper)
-  goto out_no_task;
-
  for (; nr < ARRAY_SIZE(proc_base_stuff); filp->f_pos++, nr++) {
   struct pid_entry *p = &proc_base_stuff[nr];
-  if (proc_base_fill_cache(filp, dirent, filldir, reaper, p) < 0)
+  if (proc_base_fill_cache(filp, dirent, filldir, p) < 0)
    goto out;
  }

@@ -2223,8 +2214,6 @@ int proc_pid_readdir(struct file * filp,
  }
  filp->f_pos = PID_MAX_LIMIT + TGID_OFFSET;
 out:
- put_task_struct(reaper);
-out_no_task:
  return 0;
 }

diff -puN fs/proc/root.c~funny-proc-patch fs/proc/root.c
_
```

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