Subject: [PATCH] smbfs: Make conn_pid a struct pid
Posted by ebiederm on Mon, 11 Dec 2006 13:20:22 GMT
View Forum Message <> Reply to Message

smbfs keeps track of the user space server process in conn_pid.
This converts that track to use a struct pid instead of pid_t.
This keeps us safe from pid wrap around issues and prepares the way
for the pid namespace.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>
---
 fs/smbfs/inode.c | 5 +++--
 fs/smbfs/proc.c | 6 +++---
 fs/smbfs/smbiod.c | 5 +++--
 include/linux/smb_fs_sb.h | 2 +-
 4 files changed, 10 insertions(+), 8 deletions(-)

diff --git a/fs/smbfs/inode.c b/fs/smbfs/inode.c
index 4af4cd7..84dfe3f 100644
--- a/fs/smbfs/inode.c
+++ b/fs/smbfs/inode.c
@@ -482,12 +482,13 @@ smb_put_super(struct super_block *sb)
  smb_close_socket(server);

  if (server->conn_pid)
- kill_proc(server->conn_pid, SIGTERM, 1);
+ kill_pid(server->conn_pid, SIGTERM, 1);

  kfree(server->ops);
  smb_unload_nls(server);
  sb->s_fs_info = NULL;
  smb_unlock_server(server);
+ put_pid(server->conn_pid);
  kfree(server);
 }

@@ -530,7 +531,7 @@ static int smb_fill_super(struct super_block *sb, void *raw_data, int silent)
  INIT_LIST_HEAD(&server->xmitq);
  INIT_LIST_HEAD(&server->recvq);
  server->conn_error = 0;
- server->conn_pid = 0;
+ server->conn_pid = NULL;
  server->state = CONN_INVALID; /* no connection yet */
  server->generation = 0;

diff --git a/fs/smbfs/proc.c b/fs/smbfs/proc.c
index a5ced9e..feac460 100644
--- a/fs/smbfs/proc.c

```
+++ b/fs/smbfs/proc.c
@@ -877,7 +877,7 @@ smb_newconn(struct smb_sb_info *server, struct smb_conn_opt *opt)
   goto out_putf;

  server->sock_file = filp;
- server->conn_pid = current->pid;
+ server->conn_pid = get_pid(task_pid(current));
  server->opt = *opt;
  server->generation += 1;
  server->state = CONN_VALID;
@@ -971,8 +971,8 @@ smb_newconn(struct smb_sb_info *server, struct smb_conn_opt *opt)
  }

  VERBOSE("protocol=%d, max_xmit=%d, pid=%d capabilities=0x%x\n",
-  server->opt.protocol, server->opt.max_xmit, server->conn_pid,
-  server->opt.capabilities);
+  server->opt.protocol, server->opt.max_xmit,
+  pid_nr(server->conn_pid), server->opt.capabilities);

  /* FIXME: this really should be done by smbmount. */
  if (server->opt.max_xmit > SMB_MAX_PACKET_SIZE) {
diff --git a/fs/smbfs/smbiod.c b/fs/smbfs/smbiod.c
index e675404..89eaf31 100644
--- a/fs/smbfs/smbiod.c
+++ b/fs/smbfs/smbiod.c
@@ -152,7 +152,7 @@ int smbiod_retry(struct smb_sb_info *server)
 {
  struct list_head *head;
  struct smb_request *req;
- pid_t pid = server->conn_pid;
+ struct pid *pid = get_pid(server->conn_pid);
  int result = 0;

  VERBOSE("state: %d\n", server->state);
@@ -222,7 +222,7 @@ int smbiod_retry(struct smb_sb_info *server)
  /*
   * Note: use the "priv" flag, as a user process may need to reconnect.
   */
- result = kill_proc(pid, SIGUSR1, 1);
+ result = kill_pid(pid, SIGUSR1, 1);
  if (result) {
   /* FIXME: this is most likely fatal, umount? */
   printk(KERN_ERR "smb_retry: signal failed [%d]\n", result);
@@ -233,6 +233,7 @@ int smbiod_retry(struct smb_sb_info *server)
  /* FIXME: The retried requests should perhaps get a "time boost". */

 out:
+ put_pid(pid);
```

```
  return result;
 }

diff --git a/include/linux/smb_fs_sb.h b/include/linux/smb_fs_sb.h
index 5b4ae2c..3aa97aa 100644
--- a/include/linux/smb_fs_sb.h
+++ b/include/linux/smb_fs_sb.h
@@ -55,7 +55,7 @@ struct smb_sb_info {
    * generation is incremented.
    */
   unsigned int generation;
- pid_t conn_pid;
+ struct pid *conn_pid;
   struct smb_conn_opt opt;
   wait_queue_head_t conn_wq;
   int conn_complete;
--
1.4.4.1.g278f
```

_____

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