
Subject: [PATCH 8/28] [PREP 8/14] Make proc be able to have multiple super blocks

Posted by [Pavel Emelianov](#) on Fri, 15 Jun 2007 16:06:30 GMT

[View Forum Message](#) <> [Reply to Message](#)

Each namespace must have its own proc super block where dentries and inodes representing the tasks live. Plus proc super block must hold the namespace it draws the pids from.

Making one proc mount with filtering looks very ugly for the following reasons. Two namespaces can have one numerical id refer to different tasks, e.g. 1 in init namespace refers to init, but 1 in sub namespace does not. This creates difficulties in proc. However having many proc trees solves this problem and prepares the ground for the proc virtualization.

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

```
fs/proc/inode.c      | 20 ++++++++----
fs/proc/internal.h  |  2 +
fs/proc/root.c      | 75 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
include/linux/proc_fs.h |  3 +
4 files changed, 91 insertions(+), 9 deletions(-)
```

--- ./fs/proc/inode.c.procmount 2007-06-15 15:00:32.000000000 +0400

+++ ./fs/proc/inode.c 2007-06-15 15:04:52.000000000 +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>

@@ -429,9 +430,17 @@ 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 pid_namespace *ns)

{

struct inode * root_inode;

+ struct proc_dir_entry * root_dentry;

+

+ root_dentry = &proc_root;

+ if (ns != &init_pid_ns) {

```

+ root_dentry = create_proc_root();
+ if (root_dentry == NULL)
+ goto out_no_de;
+ }

s->s_flags |= MS_NODIRATIME | MS_NOSUID | MS_NOEXEC;
s->s_blocksize = 1024;
@@ -440,8 +449,8 @@ int proc_fill_super(struct super_block *
s->s_op = &proc_sops;
s->s_time_gran = 1;

- de_get(&proc_root);
- root_inode = proc_get_inode(s, PROC_ROOT_INO, &proc_root);
+ de_get(root_dentry);
+ root_inode = proc_get_inode(s, PROC_ROOT_INO, root_dentry);
if (!root_inode)
goto out_no_root;
root_inode->i_uid = 0;
@@ -452,9 +461,10 @@ int proc_fill_super(struct super_block *
return 0;

out_no_root:
- printk("proc_read_super: get root inode failed\n");
iput(root_inode);
- de_put(&proc_root);
+ de_put(root_dentry);
+out_no_de:
+ printk("proc_read_super: get root inode failed\n");
return -ENOMEM;
}
MODULE_LICENSE("GPL");
--- ./fs/proc/internal.h.procmount 2007-06-15 15:00:32.000000000 +0400
+++ ./fs/proc/internal.h 2007-06-15 15:04:52.000000000 +0400
@@ -71,3 +71,5 @@ static inline int proc_fd(struct inode *
{
return PROC_I(inode)->fd;
}
+
+struct proc_dir_entry * create_proc_root(void);
--- ./fs/proc/root.c.procmount 2007-06-15 15:00:32.000000000 +0400
+++ ./fs/proc/root.c 2007-06-15 15:05:16.000000000 +0400
@@ -18,32 +18,79 @@
#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;
+ get_pid_ns(ns);
+ sb->s_fs_info = data;
+ 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);
+
+ 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, ns);
+ if (err) {

```

```

+ up_write(&sb->s_umount);
+ deactivate_super(sb);
+ return err;
+ }
+
+ ei = PROC_I(sb->s_root->d_inode);
+ if (!ei->pid)
+ ei->pid = find_get_pid(1);
+ sb->s_flags |= MS_ACTIVE;
+ }
+
+ return simple_set_mnt(mnt, sb);
+}
+
+static void proc_kill_sb(struct super_block *sb)
+{
+ put_pid_ns((struct pid_namespace *)sb->s_fs_info);
+ kill_anon_super(sb);
+ }

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)
@@ -153,6 +200,28 @@ struct proc_dir_entry proc_root = {
.parent = &proc_root,
};

+/*
+ * creates the proc root entry for different proc trees
+ */
+
+struct proc_dir_entry * create_proc_root(void)
+{
+ struct proc_dir_entry *de;
+
+ de = kzalloc(sizeof(struct proc_dir_entry), GFP_KERNEL);
+ if (de != NULL) {
+ de->low_ino = PROC_ROOT_INO;
+ de->namelen = 5;
+ de->name = "/proc";
+ de->mode = S_IFDIR | S_IRUGO | S_IXUGO;
+ de->nlink = 2;
+ de->proc_iops = &proc_root_inode_operations;

```

```
+ de->proc_fops = &proc_root_operations;
+ de->parent = de;
+ }
+ return de;
+}
+
EXPORT_SYMBOL(proc_symlink);
EXPORT_SYMBOL(proc_mkdir);
EXPORT_SYMBOL(create_proc_entry);
--- ./include/linux/proc_fs.h 2007-06-15 15:03:10.000000000 +0400
+++ ./include/linux/proc_fs.h 2007-06-15 15:04:52.000000000 +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 *, struct pid_namespace *);
extern struct inode *proc_get_inode(struct super_block *, unsigned int, struct proc_dir_entry *);

/*
```

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