
Subject: [RFC][PATCH 1/4] seed /proc root inode in proc_fill_super()

Posted by [Dave Hansen](#) on Fri, 26 Jan 2007 22:42:09 GMT

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

Here's a set of patches which I believe allows multiple instances of /proc to exist. We want these because it allows each pid_namespace to have a completely different view of /proc.

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

In order to try to get multiple /proc mounts, I'm attempting to kill this variable. But, I noticed that we have all of the data to fill in the inode's pid in proc_fill_super(), where it takes a wee bit more work in proc_get_sb().

Any reason not to just do it in proc_fill_super()?

```
lxc-dave/fs/proc/inode.c | 9 ++++++++
lxc-dave/fs/proc/root.c | 11 -----
2 files changed, 9 insertions(+), 11 deletions(-)
```

```
diff -puN fs/proc/inode.c~A0-seed-proc-differently fs/proc/inode.c
--- lxc/fs/proc/inode.c~A0-seed-proc-differently 2007-01-26 14:29:16.000000000 -0800
+++ lxc-dave/fs/proc/inode.c 2007-01-26 14:29:16.000000000 -0800
@@ -183,6 +183,7 @@ 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 +201,14 @@ 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~A0-seed-proc-differently fs/proc/root.c
--- lxc/fs/proc/root.c~A0-seed-proc-differently 2007-01-26 14:29:16.000000000 -0800
+++ lxc-dave/fs/proc/root.c 2007-01-26 14:29:16.000000000 -0800
@@ -30,17 +30,6 @@ struct proc_dir_entry *proc_sys_root;
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);
}

```

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

Subject: [RFC][PATCH 2/4] remove proc_mnt's use or killing inodes
Posted by [Dave Hansen](#) on Fri, 26 Jan 2007 22:42:10 GMT
[View Forum Message](#) <> [Reply to Message](#)

We use proc_mnt as a shortcut to find a superblock on which to go killing /proc inodes. This will break if we ever have more than one /proc superblock. So, use the superblock list to go find each of the /proc sb's and kill inodes on each superblock.

This eliminates one more use of proc_mnt.

lxc-dave/fs/proc/generic.c | 28 ++++++-----
1 file changed, 23 insertions(+), 5 deletions(-)

```
diff -puN fs/proc/generic.c~A1-remove-proc_mnt-0 fs/proc/generic.c
--- lxc/fs/proc/generic.c~A1-remove-proc_mnt-0 2007-01-26 14:29:17.000000000 -0800
+++ lxc-dave/fs/proc/generic.c 2007-01-26 14:29:17.000000000 -0800
@@ -547,13 +547,10 @@ static int proc_register(struct proc_dir
    return 0;
}

-/*
- * Kill an inode that got unregistered..
- */
-static void proc_kill_inodes(struct proc_dir_entry *de)
+static void proc_kill_inodes_sb(struct proc_dir_entry *de,
+    struct super_block *sb)
{
    struct list_head *p;
- struct super_block *sb = proc_mnt->mnt_sb;

    /*
     * Actually it's a partial revoke().
@@ -577,6 +574,27 @@ static void proc_kill_inodes(struct proc
    file_list_unlock();
}

+/*
+ * Kill an inode that got unregistered..
+ */
+static void proc_kill_inodes(struct proc_dir_entry *de)
+{
+    struct list_head *l;
+    struct file_system_type *procfs;
+
+    procfs = get_fs_type("proc");
+    if (!procfs)
+        return;
+
+    spin_lock(&sb_lock);
+    list_for_each(l, &procfs->fs_supers) {
+        struct super_block *sb;
+        sb = list_entry(l, struct super_block, s_instances);
+        proc_kill_inodes_sb(de, sb);
+    }
+    spin_unlock(&sb_lock);
+}

```

```
+
static struct proc_dir_entry *proc_create(struct proc_dir_entry **parent,
    const char *name,
    mode_t mode,
```

—

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

Subject: [RFC][PATCH 3/4] create fill_if_new_sb() helper
Posted by [Dave Hansen](#) on Fri, 26 Jan 2007 22:42:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

As I was migrating /proc away from get_sb_single() use, I noticed I was duplicating a lot of functionality in the /proc code from get_sb_single(). I then noticed that get_sb_nodev() does basically the same thing.

So, I created a fill_if_new_sb() helper to work on the superblock after sget() to fill it only when necessary and handle the work with sb->s_flags and the s->s_umount semaphore.

I'll use this in a moment in /proc as well.

```
lxc-dave/fs/super.c      | 43 ++++++-----
lxc-dave/include/linux/fs.h | 2 ++
2 files changed, 25 insertions(+), 20 deletions(-)
```

```
diff -puN fs/super.c~A2-fill_if_new_sb fs/super.c
--- lxc/fs/super.c~A2-fill_if_new_sb 2007-01-26 14:29:17.000000000 -0800
+++ lxc-dave/fs/super.c 2007-01-26 14:29:17.000000000 -0800
@@ -810,6 +810,23 @@ void kill_block_super(struct super_block
EXPORT_SYMBOL(kill_block_super);
#endif
```

```
+int fill_if_new_sb(struct super_block *s, void *data, int flags,
+ int (*fill_super)(struct super_block *, void *, int))
+{
+ int error;
+ if (s->s_root)
+ return 0;
+ s->s_flags = flags;
+ error = fill_super(s, data, flags & MS_SILENT ? 1 : 0);
+ if (error) {
```

```

+ up_write(&s->s_umount);
+ deactivate_super(s);
+ return error;
+ }
+ s->s_flags |= MS_ACTIVE;
+ return error;
+}
+
int get_sb_nodev(struct file_system_type *fs_type,
int flags, void *data,
int (*fill_super)(struct super_block *, void *, int),
@@ -820,16 +837,9 @@ int get_sb_nodev(struct file_system_type

if (IS_ERR(s))
return PTR_ERR(s);
-
- s->s_flags = flags;
-
- error = fill_super(s, data, flags & MS_SILENT ? 1 : 0);
- if (error) {
- up_write(&s->s_umount);
- deactivate_super(s);
+ error = fill_if_new_sb(s, data, flags, fill_super);
+ if (error)
return error;
- }
- s->s_flags |= MS_ACTIVE;
return simple_set_mnt(mnt, s);
}

@@ -845,22 +855,15 @@ int get_sb_single(struct file_system_type
int (*fill_super)(struct super_block *, void *, int),
struct vfsmount *mnt)
{
- struct super_block *s;
- int error;
+ struct super_block *s;

s = sget(fs_type, compare_single, set_anon_super, NULL);
if (IS_ERR(s))
return PTR_ERR(s);
- if (!s->s_root) {
- s->s_flags = flags;
- error = fill_super(s, data, flags & MS_SILENT ? 1 : 0);
- if (error) {
- up_write(&s->s_umount);
- deactivate_super(s);
- return error;

```

```

- }
- s->s_flags |= MS_ACTIVE;
- }
+ error = fill_if_new_sb(s, data, flags, fill_super);
+ if (error)
+ return error;
+ do_remount_sb(s, flags, data, 0);
+ return simple_set_mnt(mnt, s);
+ }
diff -puN include/linux/fs.h~A2-fill_if_new_sb include/linux/fs.h
--- lxc/include/linux/fs.h~A2-fill_if_new_sb 2007-01-26 14:29:17.000000000 -0800
+++ lxc-dave/include/linux/fs.h 2007-01-26 14:29:17.000000000 -0800
@@ -1416,6 +1416,8 @@ extern int get_sb_nodev(struct file_syst
    int flags, void *data,
    int (*fill_super)(struct super_block *, void *, int),
    struct vfsmount *mnt);
+extern int fill_if_new_sb(struct super_block *s, void *data, int flags,
+ int (*fill_super)(struct super_block *, void *, int));
void generic_shutdown_super(struct super_block *sb);
void kill_block_super(struct super_block *sb);
void kill_anon_super(struct super_block *sb);

```

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

Subject: [RFC][PATCH 4/4] introduce proc_mnt for pid_ns
Posted by [Dave Hansen](#) on Fri, 26 Jan 2007 22:42:12 GMT
[View Forum Message](#) <> [Reply to Message](#)

The following patch completes the removal of 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.

```

lxc-dave/fs/proc/base.c      | 26 ++++++
lxc-dave/fs/proc/inode.c     | 2 -
lxc-dave/fs/proc/root.c      | 35 ++++++

```

```
lxc-dave/include/linux/pid_namespace.h | 1
lxc-dave/include/linux/proc_fs.h | 1
5 files changed, 53 insertions(+), 12 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 2007-01-26 14:29:18.000000000 -0800
+++ lxc-dave/fs/proc/base.c 2007-01-26 14:29:18.000000000 -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 2007-01-26 14:29:18.000000000 -0800
+++ lxc-dave/fs/proc/inode.c 2007-01-26 14:29:18.000000000 -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;
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 2007-01-26 14:29:18.000000000 -0800
+++ lxc-dave/fs/proc/root.c 2007-01-26 14:29:18.000000000 -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"

@@ -27,10 +28,36 @@ 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)
{
- 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 = {
@@ -47,12 +74,6 @@ 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 2007-01-26 14:29:18.000000000
-0800

```

```

+++ lxc-dave/include/linux/pid_namespace.h 2007-01-26 14:29:18.000000000 -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 2007-01-26 14:29:18.000000000 -0800
+++ lxc-dave/include/linux/proc_fs.h 2007-01-26 14:29:18.000000000 -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 *);

```

—

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

Subject: Re: [RFC][PATCH 1/4] seed /proc root inode in proc_fill_super()
Posted by [ebiederm](#) on Sun, 28 Jan 2007 20:09:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen <hansendc@us.ibm.com> writes:

```

> Here's a set of patches which I believe allows multiple
> instances of /proc to exist. We want these because it
> allows each pid_namespace to have a completely different
> view of /proc.
>
> ---
>
> /proc currently has some special code to make sure
> that the root directory gets set up correctly. It
> uses the global proc_mnt variable in order to find
> its way to the root inode.
>
> In order to try to get multiple /proc mounts, I'm
> attempting to kill this variable. But, I noticed
> that we have all of the data to fill in the inode's

```

> pid in proc_fill_super(), where it takes a wee bit
> more work in proc_get_sb().
>
> Any reason not to just do it in proc_fill_super()?

Because we are using get_sb_single. So proc_fill_super is only called once when proc_mnt is established.

Since proc_mnt is created before pid == 1 exists, ei->pid on the root_inode is never filled in.

Once proc_mnt disappears we can make this change, but it should definitely not come first in the patch set or horrible things will happen to git-bisect users.

Eric

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

Subject: Re: [RFC][PATCH 4/4] introduce proc_mnt for pid_ns
Posted by [Sukadev Bhattiprolu](#) on Fri, 02 Feb 2007 04:17:05 GMT
[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen [hansendc@us.ibm.com] wrote:

```
|  
| The following patch completes the removal of 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.  
|  
| ---  
|  
| lxc-dave/fs/proc/base.c          | 26 ++++++++  
| lxc-dave/fs/proc/inode.c         | 2 -  
| lxc-dave/fs/proc/root.c          | 35 ++++++++  
| lxc-dave/include/linux/pid_namespace.h | 1  
| lxc-dave/include/linux/proc_fs.h   | 1  
| 5 files changed, 53 insertions(+), 12 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 2007-01-26 14:29:18.000000000 -0800
| +++ lxc-dave/fs/proc/base.c 2007-01-26 14:29:18.000000000 -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 2007-01-26 14:29:18.000000000 -0800
+++ lxc-dave/fs/proc/inode.c 2007-01-26 14:29:18.000000000 -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;
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 2007-01-26 14:29:18.000000000 -0800
+++ lxc-dave/fs/proc/root.c 2007-01-26 14:29:18.000000000 -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"
|
@@ -27,10 +28,36 @@ 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)
| {
| - 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 = {
| @@ -47,12 +74,6 @@ 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;
| - }

```

If we remove the kern_mount() call here, where do we mount /proc for the init_pid_ns now ? i.e how does proc_get_sb() get called the first time ?

```

| 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 2007-01-26 14:29:18.000000000
-0800
| +++ lxc-dave/include/linux/pid_namespace.h 2007-01-26 14:29:18.000000000 -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 2007-01-26 14:29:18.000000000 -0800
| +++ lxc-dave/include/linux/proc_fs.h 2007-01-26 14:29:18.000000000 -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 *);
|
| _
```

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

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