
Subject: [PATCH 4/15] Make proc_flush_task() flush entries from multiple proc trees
Posted by [Pavel Emelianov](#) on Thu, 26 Jul 2007 14:48:59 GMT
[View Forum Message](#) <> [Reply to Message](#)

Since a task will appear in more than one proc tree we need to shrink many trees. For this case we pass the struct pid to proc_flush_task() and shrink the mounts of all the namespaces this pid belongs to.

The NULL passed to it means that only global mount is to be flushed.
This is a preparation for a real flushing patch.

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

```
fs/proc/base.c      | 18 ++++++++-----
include/linux/proc_fs.h | 6 ++++--
kernel/exit.c       | 2 +-
3 files changed, 20 insertions(+), 6 deletions(-)
```

```
diff -upr linux-2.6.23-rc1-mm1.orig/fs/proc/base.c linux-2.6.23-rc1-mm1-7/fs/proc/base.c
--- linux-2.6.23-rc1-mm1.orig/fs/proc/base.c 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/fs/proc/base.c 2007-07-26 16:36:37.000000000 +0400
```

```
@@ -74,6 +74,7 @@
```

```
#include <linux/nsproxy.h>
```

```
#include <linux/oom.h>
```

```
#include <linux/elf.h>
```

```
+#include <linux/pid_namespace.h>
```

```
#include "internal.h"
```

```
/* NOTE:
```

```
@@ -2115,7 +2116,7 @@ static const struct inode_operations pro
```

```
* 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)
```

```
+static void proc_flush_task_mnt(struct task_struct *task, struct vfsmount *mnt)
```

```
{
    struct dentry *dentry, *leader, *dir;
```

```
    char buf[PROC_NUMBUF];
```

```
@@ -2123,7 +2124,7 @@ void proc_flush_task(struct task_struct
```

```
    name.name = buf;
```

```
    name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
```

```
- dentry = d_hash_and_lookup(proc_mnt->mnt_root, &name);
```

```
+ dentry = d_hash_and_lookup(mnt->mnt_root, &name);
```

```
    if (dentry) {
```

```
        shrink_dcache_parent(dentry);
```

```

    d_drop(dentry);
@@ -2135,7 +2136,7 @@ void proc_flush_task(struct task_struct

    name.name = buf;
    name.len = snprintf(buf, sizeof(buf), "%d", task->tgid);
- leader = d_hash_and_lookup(proc_mnt->mnt_root, &name);
+ leader = d_hash_and_lookup(mnt->mnt_root, &name);
    if (!leader)
        goto out;

@@ -2161,6 +2162,17 @@ out:
    return;
}

+/*
+ * when flushing dentries from proc one need to flush them from global
+ * proc (proc_mnt) and from all the namespaces' procs this task was seen
+ * in. this call is supposed to make all this job.
+ */
+
+void proc_flush_task(struct task_struct *task, struct pid *pid)
+{
+    proc_flush_task_mnt(task, proc_mnt);
+}
+
static struct dentry *proc_pid_instantiate(struct inode *dir,
    struct dentry * dentry,
    struct task_struct *task, const void *ptr)
diff -upr linux-2.6.23-rc1-mm1.orig/include/linux/proc_fs.h
linux-2.6.23-rc1-mm1-7/include/linux/proc_fs.h
--- linux-2.6.23-rc1-mm1.orig/include/linux/proc_fs.h 2007-07-26 16:34:45.000000000 +0400
+++ linux-2.6.23-rc1-mm1-7/include/linux/proc_fs.h 2007-07-26 16:36:36.000000000 +0400
@@ -111,7 +111,7 @@ extern void proc_misc_init(void);

struct mm_struct;

-void proc_flush_task(struct task_struct *task);
+void proc_flush_task(struct task_struct *task, struct pid *pid);
struct dentry *proc_pid_lookup(struct inode *dir, struct dentry * dentry, struct nameidata *);
int proc_pid_readdir(struct file * filp, void * dirent, filldir_t filldir);
unsigned long task_vsize(struct mm_struct *);
@@ -223,7 +227,9 @@ static inline void proc_net_remove(const
#define proc_net_create(name, mode, info) ({ (void)(mode), NULL; })
static inline void proc_net_remove(const char *name) {}

-static inline void proc_flush_task(struct task_struct *task) { }
+static inline void proc_flush_task(struct task_struct *task, struct pid *pid)
+{

```

+}

```
static inline struct proc_dir_entry *create_proc_entry(const char *name,  
    mode_t mode, struct proc_dir_entry *parent) { return NULL; }  
diff -upr linux-2.6.23-rc1-mm1.orig/kernel/exit.c linux-2.6.23-rc1-mm1-7/kernel/exit.c  
--- linux-2.6.23-rc1-mm1.orig/kernel/exit.c 2007-07-26 16:34:45.000000000 +0400  
+++ linux-2.6.23-rc1-mm1-7/kernel/exit.c 2007-07-26 16:36:37.000000000 +0400  
@@ -184,7 +199,7 @@ repeat:  
    }
```

```
    write_unlock_irq(&tasklist_lock);  
- proc_flush_task(p);  
+ proc_flush_task(p, NULL);  
    release_thread(p);  
    call_rcu(&p->rcu, delayed_put_task_struct);
```
