

---

Subject: [PATCH 5/20] Prepare proc\_flush\_task() to flush entries from multiple proc trees

Posted by [Pavel Emelianov](#) on Tue, 07 Aug 2007 09:29:44 GMT

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

---

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

---

```
fs/proc/base.c      | 27 ++++++-----
include/linux/proc_fs.h | 4 +++-
kernel/exit.c       | 2 +-
3 files changed, 24 insertions(+), 9 deletions(-)
```

--- ./fs/proc/base.c.ve5 2007-08-06 12:44:54.000000000 +0400

+++ ./fs/proc/base.c 2007-08-06 12:44:56.000000000 +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:
```

```
@@ -2115,27 +2116,27 @@ 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 vfsmount *mnt, pid_t pid, pid_t tgid)
```

```
{
    struct dentry *dentry, *leader, *dir;
    char buf[PROC_NUMBUF];
    struct qstr name;
```

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

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

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

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

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

```
if (dentry) {
    shrink_dcache_parent(dentry);
    d_drop(dentry);
    dput(dentry);
}
```

```
- if (thread_group_leader(task))
```

```
+ if (tgid == 0)
```

```

goto out;

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

@@ -2146,7 +2147,7 @@ void proc_flush_task(struct task_struct
  goto out_put_leader;

name.name = buf;
- name.len = snprintf(buf, sizeof(buf), "%d", task->pid);
+ name.len = snprintf(buf, sizeof(buf), "%d", pid);
  dentry = d_hash_and_lookup(dir, &name);
  if (dentry) {
    shrink_dcache_parent(dentry);
@@ -2161,6 +2162,18 @@ 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)
+{
+  proc_flush_task_mnt(proc_mnt, task->pid,
+  thread_group_leader(task) ? 0 : task->tgid);
+}
+
static struct dentry *proc_pid_instantiate(struct inode *dir,
    struct dentry * dentry,
    struct task_struct *task, const void *ptr)
--- ./include/linux/proc_fs.h.ve5 2007-08-06 12:44:54.000000000 +0400
+++ ./include/linux/proc_fs.h 2007-08-06 12:44:56.000000000 +0400
@@ -223,7 +223,9 @@ 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)
+{
+}

```

```

static inline struct proc_dir_entry *create_proc_entry(const char *name,
    mode_t mode, struct proc_dir_entry *parent) { return NULL; }
--- ./kernel/exit.c.ve5 2007-08-06 12:44:56.000000000 +0400
+++ ./kernel/exit.c 2007-08-06 12:44:56.000000000 +0400
@@ -157,6 +157,7 @@ void release_task(struct task_struct * p
    int zap_leader;
repeat:
    atomic_dec(&p->user->processes);
+ proc_flush_task(p);
    write_lock_irq(&tasklist_lock);
    ptrace_unlink(p);
    BUG_ON(!list_empty(&p->ptrace_list) || !list_empty(&p->ptrace_children));
@@ -184,7 +185,6 @@ repeat:
    }

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

```

---