Subject: [PATCH 4/20] Prepare proc_flust_task() to flush entries from multiple proc trees
Posted by Pavel Emelianov on Fri, 10 Aug 2007 11:47:57 GMT
View Forum Message <> Reply to Message

The first part is trivial - we just make the proc_flush_task() to
operate on arbitrary vfsmount with arbitrary ids and pass the pid and
global proc_mnt to it.

The other change is more tricky: I moved the proc_flush_task() call in
release_task() higher to address the following problem.

When flushing task from many proc trees we need to know the set of ids
(not just one pid) to find the dentries' names to flush. Thus we need
to pass the task's pid to proc_flush_task() as struct pid is the only
object that can provide all the pid numbers. But after __exit_signal()
task has detached all his pids and this information is lost.

This creates a tiny gap for proc_pid_lookup() to bring some dentries
back to tree and keep them in hash (since pids are still alive before
__exit_signal()) till the next shrink, but since proc_flush_task()
does not provide a 100% guarantee that the dentries will be flushed,
this is OK to do so.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>
Cc: Oleg Nesterov <oleg@tv-sign.ru>

---

 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);
```