Subject: [PATCH] proc: fix PDE refcounting
Posted by Alexey Dobriyan on Fri, 30 Nov 2007 13:05:02 GMT
View Forum Message <> Reply to Message

Creating PDEs with refcount 0 and "deleted" flag has problems (see below).
Switch to usual scheme:
* PDE is created with refcount 1
* every de_get does +1
* every de_put() and remove_proc_entry() do -1
* once refcount reaches 0, PDE is freed.

This elegantly fixes at least two following races (both observed) without
introducing new locks, without abusing old locks, without spreading
lock_kernel():

1) PDE leak

```
remove_proc_entry   de_put
-----------------   ------
   [refcnt = 1]
if (atomic_read(&de->count) == 0)
     if (atomic_dec_and_test(&de->count))
      if (de->deleted)
       /* also not taken! */
       free_proc_entry(de);
else
 de->deleted = 1;
  [refcount=0, deleted=1]
```

2) use after free

```
remove_proc_entry   de_put
-----------------   ------
   [refcnt = 1]

     if (atomic_dec_and_test(&de->count))
if (atomic_read(&de->count) == 0)
 free_proc_entry(de);
      /* boom! */
      if (de->deleted)
       free_proc_entry(de);
```

BUG: unable to handle kernel paging request at virtual address 6b6b6b6b
printing eip: c10acdda *pdpt = 00000000338f8001 *pde = 0000000000000000
Oops: 0000 [#1] PREEMPT SMP
Modules linked in: af_packet ipv6 cpufreq_ondemand loop serio_raw psmouse k8temp hwmon sr_mod cdrom
Pid: 23161, comm: cat Not tainted (2.6.24-rc2-8c0863403f109a43d7000b4646da4818220d501f

#4)
EIP: 0060:[<c10acdda>] EFLAGS: 00210097 CPU: 1
EIP is at strnlen+0x6/0x18
EAX: 6b6b6b6b EBX: 6b6b6b6b ECX: 6b6b6b6b EDX: fffffffe
ESI: c128fa3b EDI: f380bf34 EBP: ffffffff ESP: f380be44
 DS: 007b ES: 007b FS: 00d8 GS: 0033 SS: 0068
Process cat (pid: 23161, ti=f380b000 task=f38f2570 task.ti=f380b000)
Stack: c10ac4f0 00000278 c12ce000 f43cd2a8 00000163 00000000 7da86067 00000400
    c128fa20 00896b18 f38325a8 c128fe20 ffffffff 00000000 c11f291e 00000400
    f75be300 c128fa20 f769c9a0 c10ac779 f380bf34 f7bfee70 c1018e6b f380bf34
Call Trace:
 [<c10ac4f0>] vsnprintf+0x2ad/0x49b
 [<c10ac779>] vscnprintf+0x14/0x1f
 [<c1018e6b>] vprintk+0xc5/0x2f9
 [<c10379f1>] handle_fasteoi_irq+0x0/0xab
 [<c1004f44>] do_IRQ+0x9f/0xb7
 [<c117db3b>] preempt_schedule_irq+0x3f/0x5b
 [<c100264e>] need_resched+0x1f/0x21
 [<c10190ba>] printk+0x1b/0x1f
 [<c107c8ad>] de_put+0x3d/0x50
 [<c107c8f8>] proc_delete_inode+0x38/0x41
 [<c107c8c0>] proc_delete_inode+0x0/0x41
 [<c1066298>] generic_delete_inode+0x5e/0xc6
 [<c1065aa9>] iput+0x60/0x62
 [<c1063c8e>] d_kill+0x2d/0x46
 [<c1063fa9>] dput+0xdc/0xe4
 [<c10571a1>] __fput+0xb0/0xcd
 [<c1054e49>] filp_close+0x48/0x4f
 [<c1055ee9>] sys_close+0x67/0xa5
 [<c10026b6>] sysenter_past_esp+0x5f/0x85
=======================
Code: c9 74 0c f2 ae 74 05 bf 01 00 00 00 4f 89 fa 5f 89 d0 c3 85 c9 57 89 c7 89 d0 74 05 f2 ae 75 01 4f 89 f8 5f c3 89 c1 89 c8 eb 06 <80> 38 00 74 07 40 4a 83 fa ff 75 f4 29 c8 c3 90 90 90 57 83 c9
EIP: [<c10acdda>] strnlen+0x6/0x18 SS:ESP 0068:f380be44

Also, remove broken usage of ->deleted from reiserfs: if sget() succeeds, module is already pinned and remove_proc_entry() can't happen => nobody can mark PDE deleted.

Dummy proc root in netns code is not marked with refcount 1. AFAICS, we never get it, it's just for proper /proc/net removal. I double checked CLONE_NETNS continues to work.

Patch survives many hours of modprobe/rmmod/cat loops without new bugs which can be attributed to refcounting.

Signed-off-by: Alexey Dobriyan <adobriyan@sw.ru>

```
 fs/proc/generic.c       |    9 ++-------
 fs/proc/inode.c         |    9 ++-------
 fs/proc/root.c          |    1 +
 fs/reiserfs/procfs.c    |    6 ------
 include/linux/proc_fs.h |    1 -
 5 files changed, 5 insertions(+), 21 deletions(-)

--- a/fs/proc/generic.c
+++ b/fs/proc/generic.c
@@ -585,6 +585,7 @@ static struct proc_dir_entry *proc_create(struct proc_dir_entry **parent,
  ent->namelen = len;
  ent->mode = mode;
  ent->nlink = nlink;
+ atomic_set(&ent->count, 1);
  ent->pde_users = 0;
  spin_lock_init(&ent->pde_unload_lock);
  ent->pde_unload_completion = NULL;
@@ -682,7 +683,6 @@ void free_proc_entry(struct proc_dir_entry *de)

 /*
  * Remove a /proc entry and free it if it's not currently in use.
- * If it is in use, we set the 'deleted' flag.
  */
 void remove_proc_entry(const char *name, struct proc_dir_entry *parent)
 {
@@ -731,13 +731,8 @@ continue_removing:
    parent->nlink--;
   de->nlink = 0;
   WARN_ON(de->subdir);
-  if (!atomic_read(&de->count))
+  if (atomic_dec_and_test(&de->count))
    free_proc_entry(de);
-  else {
-   de->deleted = 1;
-   printk("remove_proc_entry: %s/%s busy, count=%d\n",
-    parent->name, de->name, atomic_read(&de->count));
-  }
   break;
  }
  spin_unlock(&proc_subdir_lock);
--- a/fs/proc/inode.c
+++ b/fs/proc/inode.c
@@ -43,13 +43,8 @@ void de_put(struct proc_dir_entry *de)
    return;
   }
```

```
-  if (atomic_dec_and_test(&de->count)) {
-   if (de->deleted) {
-    printk("de_put: deferred delete of %s\n",
-     de->name);
-    free_proc_entry(de);
-   }
-  }
+  if (atomic_dec_and_test(&de->count))
+   free_proc_entry(de);
   unlock_kernel();
  }
 }
--- a/fs/proc/root.c
+++ b/fs/proc/root.c
@@ -207,6 +207,7 @@ struct proc_dir_entry proc_root = {
  .name  = "/proc",
  .mode  = S_IFDIR | S_IRUGO | S_IXUGO,
  .nlink  = 2,
+ .count  = ATOMIC_INIT(1),
  .proc_iops = &proc_root_inode_operations,
  .proc_fops = &proc_root_operations,
  .parent  = &proc_root,
--- a/fs/reiserfs/procfs.c
+++ b/fs/reiserfs/procfs.c
@@ -420,12 +420,6 @@ static void *r_start(struct seq_file *m, loff_t * pos)
   return NULL;

  up_write(&s->s_umount);
-
- if (de->deleted) {
-  deactivate_super(s);
-  return NULL;
- }
-
  return s;
 }

--- a/include/linux/proc_fs.h
+++ b/include/linux/proc_fs.h
@@ -75,7 +75,6 @@ struct proc_dir_entry {
  read_proc_t *read_proc;
  write_proc_t *write_proc;
  atomic_t count;  /* use count */
- int deleted;  /* delete flag */
  int pde_users; /* number of callers into module in progress */
  spinlock_t pde_unload_lock; /* proc_fops checks and pde_users bumps */
  struct completion *pde_unload_completion;
```