Subject: [PATCH -mm] sn2: use static ->proc_fops
Posted by adobriyan on Mon, 29 Jan 2007 12:23:12 GMT
View Forum Message <> Reply to Message

fix-rmmod-read-write-races-in-proc-entries.patch doesn't want dynamically allocated ->proc_fops, because it will set it to NULL at module unload time.

Regardless of module status, switch to statically allocated ->proc_fops which leads to simpler code without wrappers.

AFAICS, also fix the following bug: "sn_force_interrupt" proc entry set ->write for itself, but was created with 0444 permissions. Change to 0644.

Signed-off-by: Alexey Dobriyan <adobriyan@openvz.org>

```
 arch/ia64/sn/kernel/sn2/sn_proc_fs.c |  105 +++++++++++++++++++++----------------
 1 file changed, 62 insertions(+), 43 deletions(-)

--- a/arch/ia64/sn/kernel/sn2/sn_proc_fs.c
+++ b/arch/ia64/sn/kernel/sn2/sn_proc_fs.c
@@ -89,61 +89,80 @@ static int coherence_id_open(struct inod
  return single_open(file, coherence_id_show, NULL);
 }

-static struct proc_dir_entry
-*sn_procfs_create_entry(const char *name, struct proc_dir_entry *parent,
-   int (*openfunc)(struct inode *, struct file *),
- int (*releasefunc)(struct inode *, struct file *),
- ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *))
-{
- struct proc_dir_entry *e = create_proc_entry(name, 0444, parent);
-
- if (e) {
-  struct file_operations *f;
-
-  f = kzalloc(sizeof(*f), GFP_KERNEL);
-  if (f) {
-   f->open = openfunc;
-   f->read = seq_read;
-   f->llseek = seq_lseek;
-   f->release = releasefunc;
-   f->write = write;
-   e->proc_fops = f;
-  }
- }
-
- return e;
```

```
-}
-
 /* /proc/sgi_sn/sn_topology uses seq_file, see sn_hwperf.c */
 extern int sn_topology_open(struct inode *, struct file *);
 extern int sn_topology_release(struct inode *, struct file *);

+static const struct file_operations proc_partition_id_fops = {
+ .open  = partition_id_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+static const struct file_operations proc_system_sn_fops = {
+ .open  = system_serial_number_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+static const struct file_operations proc_license_id_fops = {
+ .open  = licenseID_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+static const struct file_operations proc_sn_force_intr_fops = {
+ .open  = sn_force_interrupt_open,
+ .read  = seq_read,
+ .write  = sn_force_interrupt_write_proc,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+static const struct file_operations proc_coherence_id_fops = {
+ .open  = coherence_id_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+static const struct file_operations proc_sn_topo_fops = {
+ .open  = sn_topology_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = sn_topology_release,
+};
```

```
+
 void register_sn_procfs(void)
 {
  static struct proc_dir_entry *sgi_proc_dir = NULL;
+ struct proc_dir_entry *pde;

  BUG_ON(sgi_proc_dir != NULL);
  if (!(sgi_proc_dir = proc_mkdir("sgi_sn", NULL)))
   return;

- sn_procfs_create_entry("partition_id", sgi_proc_dir,
-  partition_id_open, single_release, NULL);
-
- sn_procfs_create_entry("system_serial_number", sgi_proc_dir,
-  system_serial_number_open, single_release, NULL);
-
- sn_procfs_create_entry("licenseID", sgi_proc_dir,
-  licenseID_open, single_release, NULL);
-
- sn_procfs_create_entry("sn_force_interrupt", sgi_proc_dir,
-  sn_force_interrupt_open, single_release,
-  sn_force_interrupt_write_proc);
-
- sn_procfs_create_entry("coherence_id", sgi_proc_dir,
-  coherence_id_open, single_release, NULL);
-
- sn_procfs_create_entry("sn_topology", sgi_proc_dir,
-  sn_topology_open, sn_topology_release, NULL);
+ pde = create_proc_entry("partition_id", 0444, sgi_proc_dir);
+ if (pde)
+  pde->proc_fops = &proc_partition_id_fops;
+ pde = create_proc_entry("system_serial_number", 0444, sgi_proc_dir);
+ if (pde)
+  pde->proc_fops = &proc_system_sn_fops;
+ pde = create_proc_entry("licenseID", 0444, sgi_proc_dir);
+ if (pde)
+  pde->proc_fops = &proc_license_id_fops;
+ pde = create_proc_entry("sn_force_interrupt", 0644, sgi_proc_dir);
+ if (pde)
+  pde->proc_fops = &proc_sn_force_intr_fops;
+ pde = create_proc_entry("coherence_id", 0444, sgi_proc_dir);
+ if (pde)
+  pde->proc_fops = &proc_coherence_id_fops;
+ pde = create_proc_entry("sn_topology", 0444, sgi_proc_dir);
+ if (pde)
+  pde->proc_fops = &proc_sn_topo_fops;
 }
```

```
#endif /* CONFIG_PROC_FS */
```