Subject: [PATCH 56/59] sysctl: factor out sysctl_head_next from do_sysctl
Posted by ebiederm on Tue, 16 Jan 2007 16:40:01 GMT
View Forum Message <> Reply to Message

From: Eric W. Biederman <ebiederm@xmission.com> - unquoted

The current logic to walk through the list of sysctl table
headers is slightly painful and implement in a way it cannot
be used by code outside sysctl.c

I am in the process of implementing a version of the sysctl
proc support that instead of using the proc generic non-caching
monster, just uses the existing sysctl data structure as
backing store for building the dcache entries and for doing
directory reads.   To use the existing data structures however
I need a way to get at them.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
---
 include/linux/sysctl.h |    4 +++
 kernel/sysctl.c        |   57 ++++++++++++++++++++++++++++++++++++++++------------
 2 files changed, 46 insertions(+), 15 deletions(-)

diff --git a/include/linux/sysctl.h b/include/linux/sysctl.h
index 6113f3b..81ee9ea 100644
--- a/include/linux/sysctl.h
+++ b/include/linux/sysctl.h
@@ -923,6 +923,10 @@ enum
 #ifdef __KERNEL__
 #include <linux/list.h>

+/* For the /proc/sys support */
+extern struct ctl_table_header *sysctl_head_next(struct ctl_table_header *prev);
+extern void sysctl_head_finish(struct ctl_table_header *prev);
+
 extern void sysctl_init(void);

 typedef struct ctl_table ctl_table;
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 5beee1f..ca2831a 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -1066,6 +1066,42 @@ static void start_unregistering(struct ctl_table_header *p)
  list_del_init(&p->ctl_entry);
 }

+void sysctl_head_finish(struct ctl_table_header *head)
+{
```

```
+ if (!head)
+  return;
+ spin_lock(&sysctl_lock);
+ unuse_table(head);
+ spin_unlock(&sysctl_lock);
+}
+
+struct ctl_table_header *sysctl_head_next(struct ctl_table_header *prev)
+{
+ struct ctl_table_header *head;
+ struct list_head *tmp;
+ spin_lock(&sysctl_lock);
+ if (prev) {
+  tmp = &prev->ctl_entry;
+  unuse_table(prev);
+  goto next;
+ }
+ tmp = &root_table_header.ctl_entry;
+ for (;;) {
+  head = list_entry(tmp, struct ctl_table_header, ctl_entry);
+
+  if (!use_table(head))
+   goto next;
+  spin_unlock(&sysctl_lock);
+  return head;
+ next:
+  tmp = tmp->next;
+  if (tmp == &root_table_header.ctl_entry)
+   break;
+ }
+ spin_unlock(&sysctl_lock);
+ return NULL;
+}
+
 void __init sysctl_init(void)
 {
 #ifdef CONFIG_PROC_SYSCTL
@@ -1077,6 +1113,7 @@ void __init sysctl_init(void)
 int do_sysctl(int __user *name, int nlen, void __user *oldval, size_t __user *oldlenp,
         void __user *newval, size_t newlen)
 {
+ struct ctl_table_header *head;
   struct list_head *tmp;
   int error = -ENOTDIR;

@@ -1087,26 +1124,16 @@ int do_sysctl(int __user *name, int nlen, void __user *oldval, size_t
__user *ol
   if (!oldlenp || get_user(old_len, oldlenp))
```

```
    return -EFAULT;
  }
- spin_lock(&sysctl_lock);
- tmp = &root_table_header.ctl_entry;
- do {
-  struct ctl_table_header *head =
-   list_entry(tmp, struct ctl_table_header, ctl_entry);

-  if (!use_table(head))
-   continue;
-
-  spin_unlock(&sysctl_lock);
+ for (head = sysctl_head_next(NULL); head; head = sysctl_head_next(head)) {

   error = parse_table(name, nlen, oldval, oldlenp,
      newval, newlen, head->ctl_table);
-
-  spin_lock(&sysctl_lock);
-  unuse_table(head);
-  if (error != -ENOTDIR)
+  if (error != -ENOTDIR) {
+   sysctl_head_finish(head);
    break;
- } while ((tmp = tmp->next) != &root_table_header.ctl_entry);
- spin_unlock(&sysctl_lock);
+  }
+ }
  return error;
 }

--
1.4.4.1.g278f
```

_____
Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers