Subject: [PATCH 2/3] proc: sysctl: add _proc_do_string helper
Posted by Sam Vilain on Tue, 23 May 2006 01:23:01 GMT
View Forum Message <> Reply to Message

From: Sam Vilain <sam.vilain@catalyst.net.nz>

The logic in proc_do_string is worth re-using without passing in a
ctl_table structure (say, we want to calculate a pointer and pass that
in instead); pass in the two fields it uses from that structure
as explicit arguments.

```
 kernel/sysctl.c |   65 ++++++++++++++++++++++++++++++++------------------------
 1 files changed, 36 insertions(+), 29 deletions(-)

diff --git a/kernel/sysctl.c b/kernel/sysctl.c
index 618a2f8..cf053fc 100644
--- a/kernel/sysctl.c
+++ b/kernel/sysctl.c
@@ -1605,32 +1605,14 @@ static ssize_t proc_writesys(struct file
  return do_rw_proc(1, file, (char __user *) buf, count, ppos);
 }

-/**
- * proc_dostring - read a string sysctl
- * @table: the sysctl table
- * @write: %TRUE if this is a write to the sysctl file
- * @filp: the file structure
- * @buffer: the user buffer
- * @lenp: the size of the user buffer
- * @ppos: file position
- *
- * Reads/writes a string from/to the user buffer. If the kernel
- * buffer provided is not large enough to hold the string, the
- * string is truncated. The copied string is %NULL-terminated.
- * If the string is being read by the user process, it is copied
- * and a newline '\n' is added. It is truncated if the buffer is
- * not large enough.
- *
- * Returns 0 on success.
- */
-int proc_dostring(ctl_table *table, int write, struct file *filp,
-    void __user *buffer, size_t *lenp, loff_t *ppos)
+int _proc_do_string(void* data, int maxlen, int write, struct file *filp,
+      void __user *buffer, size_t *lenp, loff_t *ppos)
 {
  size_t len;
  char __user *p;
```

```
  char c;

- if (!table->data || !table->maxlen || !*lenp ||
+ if (!data || !maxlen || !*lenp ||
      (*ppos && !write)) {
   *lenp = 0;
   return 0;
@@ -1646,20 +1628,20 @@ int proc_dostring(ctl_table *table, int
     break;
    len++;
   }
-  if (len >= table->maxlen)
-   len = table->maxlen-1;
-  if(copy_from_user(table->data, buffer, len))
+  if (len >= maxlen)
+   len = maxlen-1;
+  if(copy_from_user(data, buffer, len))
    return -EFAULT;
-  ((char *) table->data)[len] = 0;
+  ((char *) data)[len] = 0;
   *ppos += *lenp;
  } else {
-  len = strlen(table->data);
-  if (len > table->maxlen)
-   len = table->maxlen;
+  len = strlen(data);
+  if (len > maxlen)
+   len = maxlen;
   if (len > *lenp)
    len = *lenp;
   if (len)
-   if(copy_to_user(buffer, table->data, len))
+   if(copy_to_user(buffer, data, len))
     return -EFAULT;
   if (len < *lenp) {
    if(put_user('\n', ((char __user *) buffer) + len))
@@ -1672,6 +1654,31 @@ int proc_dostring(ctl_table *table, int
  return 0;
 }

+/**
+ * proc_dostring - read a string sysctl
+ * @table: the sysctl table
+ * @write: %TRUE if this is a write to the sysctl file
+ * @filp: the file structure
+ * @buffer: the user buffer
+ * @lenp: the size of the user buffer
+ * @ppos: file position
```

```
+ *
+ * Reads/writes a string from/to the user buffer. If the kernel
+ * buffer provided is not large enough to hold the string, the
+ * string is truncated. The copied string is %NULL-terminated.
+ * If the string is being read by the user process, it is copied
+ * and a newline '\n' is added. It is truncated if the buffer is
+ * not large enough.
+ *
+ * Returns 0 on success.
+ */
+int proc_dostring(ctl_table *table, int write, struct file *filp,
+    void __user *buffer, size_t *lenp, loff_t *ppos)
+{
+ return _proc_do_string(table->data, table->maxlen, write, filp,
+          buffer, lenp, ppos);
+}
+
 /*
  * Special case of dostring for the UTS structure. This has locks
  * to observe. Should this be in kernel/sys.c ????
```