
Subject: [PATCH 1/3] proc: sysctl: rename proc_doutsstring to proc_do_uts_string
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>

'proc_doutsstring' is confusing; delimit the word with more underscores.

kernel/sysctl.c | 16 ++++++-----
1 files changed, 8 insertions(+), 8 deletions(-)

diff --git a/kernel/sysctl.c b/kernel/sysctl.c

index 708cdcd..618a2f8 100644

--- a/kernel/sysctl.c

+++ b/kernel/sysctl.c

@@ -141,7 +141,7 @@ #endif

static int parse_table(int __user *, int, void __user *, size_t __user *, void __user *, size_t,
ctl_table *, void **);

-static int proc_doutsstring(ctl_table *table, int write, struct file *filp,
+static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
void __user *buffer, size_t *lenp, loff_t *ppos);

static ctl_table root_table[];

@@ -244,7 +244,7 @@ static ctl_table kern_table[] = {
.data = init_uts_ns.name.sysname,
.maxlen = sizeof(init_uts_ns.name.sysname),
.mode = 0444,
- .proc_handler = &proc_doutsstring,
+ .proc_handler = &proc_do_uts_string,
.strategy = &sysctl_string,
},
{

@@ -253,7 +253,7 @@ static ctl_table kern_table[] = {
.data = init_uts_ns.name.release,
.maxlen = sizeof(init_uts_ns.name.release),
.mode = 0444,
- .proc_handler = &proc_doutsstring,
+ .proc_handler = &proc_do_uts_string,
.strategy = &sysctl_string,
},
{

@@ -262,7 +262,7 @@ static ctl_table kern_table[] = {
.data = init_uts_ns.name.version,
.maxlen = sizeof(init_uts_ns.name.version),
.mode = 0444,
- .proc_handler = &proc_doutsstring,

```

+ .proc_handler = &proc_do_uts_string,
  .strategy = &sysctl_string,
},
{
@@ -271,7 +271,7 @@ static ctl_table kern_table[] = {
  .data = init_uts_ns.name.nodename,
  .maxlen = sizeof(init_uts_ns.name.nodename),
  .mode = 0644,
- .proc_handler = &proc_doutsstring,
+ .proc_handler = &proc_do_uts_string,
  .strategy = &sysctl_string,
},
{
@@ -280,7 +280,7 @@ static ctl_table kern_table[] = {
  .data = init_uts_ns.name.domainname,
  .maxlen = sizeof(init_uts_ns.name.domainname),
  .mode = 0644,
- .proc_handler = &proc_doutsstring,
+ .proc_handler = &proc_do_uts_string,
  .strategy = &sysctl_string,
},
{
@@ -1677,7 +1677,7 @@ int proc_dostring(ctl_table *table, int
 * to observe. Should this be in kernel/sys.c ???
*/

-static int proc_doutsstring(ctl_table *table, int write, struct file *filp,
+static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
  void __user *buffer, size_t *lenp, loff_t *ppos)
{
  int r;
@@ -2256,7 +2256,7 @@ int proc_dostring(ctl_table *table, int
  return -ENOSYS;
}

-static int proc_doutsstring(ctl_table *table, int write, struct file *filp,
+static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
  void __user *buffer, size_t *lenp, loff_t *ppos)
{
  return -ENOSYS;
}

```

>From sam.vilain@catalyst.net.nz Tue May 23 13:23:01 2006
 From: Sam Vilain <sam.vilain@catalyst.net.nz>
 Subject: [PATCH 2/3] proc: sysctl: add _proc_do_string helper
 Date: Tue, 23 May 2006 13:23:01 +1200
 To: ebiederm@xmission.com (Eric W. Biederman)
 Cc: linux-kernel@vger.kernel.org, "Serge E. Hallyn" <serue@us.ibm.com>, herbert@13thfloor.at,
 dev@sw.ru, devel@openvz.org, sam@vilain.net, xemul@sw.ru, Dave Hansen

<haveblue@us.ibm.com>, Andrew Morton <akpm@osdl.org>, Cedric Le Goater
<clg@fr.ibm.com>, Alan Cox <alan@lxorguk.ukuu.org.uk>
Message-Id: <20060523012301.13531.31499.stgit@localhost.localdomain>
In-Reply-To: <20060523012300.13531.96685.stgit@localhost.localdomain>
References: <20060523012300.13531.96685.stgit@localhost.localdomain>

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 ????
```

```

>From sam.vilain@catalyst.net.nz Tue May 23 13:23:01 2006
From: Sam Vilain <sam.vilain@catalyst.net.nz>
Subject: [PATCH 3/3] proc: make UTS-related sysctls utsns aware
Date: Tue, 23 May 2006 13:23:01 +1200
To: ebiederm@xmission.com (Eric W. Biederman)
Cc: linux-kernel@vger.kernel.org, "Serge E. Hallyn" <serue@us.ibm.com>, herbert@13thfloor.at,
dev@sw.ru, devel@openvz.org, sam@vilain.net, xemul@sw.ru, Dave Hansen
<haveblue@us.ibm.com>, Andrew Morton <akpm@osdl.org>, Cedric Le Goater
<clg@fr.ibm.com>, Alan Cox <alan@lxorguk.ukuu.org.uk>
Message-Id: <20060523012301.13531.12776.stgit@localhost.localdomain>
In-Reply-To: <20060523012300.13531.96685.stgit@localhost.localdomain>
References: <20060523012300.13531.96685.stgit@localhost.localdomain>
```

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

Add a new function proc_do_utsns_string, that derives the pointer into the uts_namespace->name structure, currently based on the filename of the dentry in /proc, and calls _proc_do_string()

RFC only - not tested yet. builds, though.

```

kernel/sysctl.c | 104 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1 files changed, 104 insertions(+), 0 deletions(-)
```

```
diff --git a/kernel/sysctl.c b/kernel/sysctl.c
```

index cf053fc..37dc17f 100644

--- a/kernel/sysctl.c

+++ b/kernel/sysctl.c

@@ -141,8 +141,13 @@ #endif

```
static int parse_table(int __user *, int, void __user *, size_t __user *, void __user *, size_t,
                        ctl_table *, void **);
```

#ifndef CONFIG_UTS_NS

```
static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
                              void __user *buffer, size_t *lenp, loff_t *ppos);
```

#else

```
+static int proc_do_utsns_string(ctl_table *table, int write, struct file *filp,
```

```
+ void __user *buffer, size_t *lenp, loff_t *ppos);
```

#endif

```
static ctl_table root_table[];
```

```
static struct ctl_table_header root_table_header =
```

@@ -238,6 +243,7 @@ #endif

```
};
```

```
static ctl_table kern_table[] = {
```

#ifndef CONFIG_UTS_NS

```
{
```

```
    .ctl_name = KERN_OSTYPE,
```

```
    .procname = "ostype",
```

@@ -283,6 +289,54 @@ static ctl_table kern_table[] = {

```
    .proc_handler = &proc_do_uts_string,
```

```
    .strategy = &sysctl_string,
```

```
},
```

#else /* !CONFIG_UTS_NS */

```
+ {
```

```
+    .ctl_name = KERN_OSTYPE,
```

```
+    .procname = "ostype",
```

```
+    .data = NULL,
```

```
+    /* could maybe use __NEW_UTS_LEN here? */
```

```
+    .maxlen = sizeof(current->nsproxy->uts_ns->name.sysname),
```

```
+    .mode = 0444,
```

```
+    .proc_handler = &proc_do_utsns_string,
```

```
+    .strategy = &sysctl_string,
```

```
+ },
```

```
+ {
```

```
+    .ctl_name = KERN_OSRELEASE,
```

```
+    .procname = "osrelease",
```

```
+    .data = NULL,
```

```
+    .maxlen = sizeof(current->nsproxy->uts_ns->name.release),
```

```
+    .mode = 0444,
```

```
+    .proc_handler = &proc_do_utsns_string,
```

```
+    .strategy = &sysctl_string,
```

```

+ },
+ {
+ .ctl_name = KERN_VERSION,
+ .procname = "version",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.version),
+ .mode = 0444,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_NODENAME,
+ .procname = "hostname",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.nodename),
+ .mode = 0644,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_DOMAINNAME,
+ .procname = "domainname",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.domainname),
+ .mode = 0644,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ #endif /* !CONFIG_UTS_NS */
+ {
+ .ctl_name = KERN_PANIC,
+ .procname = "panic",
@@ -1684,6 +1738,7 @@ int proc_dostring(ctl_table *table, int
* to observe. Should this be in kernel/sys.c ???
*/

+#ifndef CONFIG_UTS_NS
static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
void __user *buffer, size_t *lenp, loff_t *ppos)
{
@@ -1700,6 +1755,55 @@ static int proc_do_uts_string(ctl_table
}
return r;
}
+#else /* !CONFIG_UTS_NS */
+static int proc_do_utsns_string(ctl_table *table, int write, struct file *filp,
+ void __user *buffer, size_t *lenp, loff_t *ppos)
+{

```

```

+ int r;
+ struct uts_namespace* uts_ns = current->nsproxy->uts_ns;
+ char* which;
+
+ /* map the filename to the pointer. perhaps it would be better
+  to put struct offset pointers in table->data ? */
+ switch (filp->f_dentry->d_name.name[3]) {
+ case 'y': /* ostype */
+   which = uts_ns->name.sysname;
+   break;
+ case 't': /* hostname */
+   which = uts_ns->name.nodename;
+   break;
+ case 'e': /* release */
+   which = uts_ns->name.release;
+   break;
+ case 's': /* version */
+   which = uts_ns->name.version;
+   break;
+ case 'x': /* XXX - unreachable */
+   which = uts_ns->name.machine;
+   break;
+ case 'a': /* domainname */
+   which = uts_ns->name.domainname;
+   break;
+ default:
+   printk("procfs: impossible uts part '%s'",
+          (char*)filp->f_dentry->d_name.name);
+   r = -EINVAL;
+   goto out;
+ }
+
+ if (!write) {
+   down_read(&uts_sem);
+   r = _proc_do_string(which, table->maxlen, 0, filp, buffer, lenp, ppos);
+   up_read(&uts_sem);
+ } else {
+   down_write(&uts_sem);
+   r = _proc_do_string(which, table->maxlen, 1, filp, buffer, lenp, ppos);
+   up_write(&uts_sem);
+ }
+ out:
+ return r;
+}
+#endif /* !CONFIG_UTS_NS */

```

```

static int do_proc_dointvec_conv(int *negp, unsigned long *lvalp,
    int *valp,

```

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 ???

```

Subject: [PATCH 3/3] proc: make UTS-related sysctls utsns aware

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>

Add a new function proc_do_utsns_string, that derives the pointer into the uts_namespace->name structure, currently based on the filename of the dentry in /proc, and calls _proc_do_string()

RFC only - not tested yet. builds, though.

```

kernel/sysctl.c | 104 +++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1 files changed, 104 insertions(+), 0 deletions(-)

```

```

diff --git a/kernel/sysctl.c b/kernel/sysctl.c

```

```

index cf053fc..37dc17f 100644

```

```

--- a/kernel/sysctl.c

```

```

+++ b/kernel/sysctl.c

```

```

@@ -141,8 +141,13 @@ #endif

```

```

static int parse_table(int __user *, int, void __user *, size_t __user *, void __user *, size_t,
      ctl_table *, void **);
+#ifndef CONFIG_UTS_NS
static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
      void __user *buffer, size_t *lenp, loff_t *ppos);
+#else

```

```

+static int proc_do_utsns_string(ctl_table *table, int write, struct file *filp,
+ void __user *buffer, size_t *lenp, loff_t *ppos);
+endif

static ctl_table root_table[];
static struct ctl_table_header root_table_header =
@@ -238,6 +243,7 @@ #endif
};

static ctl_table kern_table[] = {
#ifdef CONFIG_UTS_NS
{
    .ctl_name = KERN_OSTYPE,
    .procname = "ostype",
@@ -283,6 +289,54 @@ static ctl_table kern_table[] = {
    .proc_handler = &proc_do_uts_string,
    .strategy = &sysctl_string,
},
#else /* !CONFIG_UTS_NS */
+ {
+ .ctl_name = KERN_OSTYPE,
+ .procname = "ostype",
+ .data = NULL,
+ /* could maybe use __NEW_UTS_LEN here? */
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.sysname),
+ .mode = 0444,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_OSRELEASE,
+ .procname = "osrelease",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.release),
+ .mode = 0444,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_VERSION,
+ .procname = "version",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.version),
+ .mode = 0444,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {

```

```

+ .ctl_name = KERN_NODENAME,
+ .procname = "hostname",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.nodename),
+ .mode = 0644,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_DOMAINNAME,
+ .procname = "domainname",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.domainname),
+ .mode = 0644,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ #endif /* !CONFIG_UTS_NS */
+ {
+ .ctl_name = KERN_PANIC,
+ .procname = "panic",
@@ -1684,6 +1738,7 @@ int proc_dostring(ctl_table *table, int
* to observe. Should this be in kernel/sys.c ???
*/

+ #ifndef CONFIG_UTS_NS
+ static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
+ void __user *buffer, size_t *lenp, loff_t *ppos)
+ {
@@ -1700,6 +1755,55 @@ static int proc_do_uts_string(ctl_table
+ }
+ return r;
+ }
+ #else /* !CONFIG_UTS_NS */
+ static int proc_do_utsns_string(ctl_table *table, int write, struct file *filp,
+ void __user *buffer, size_t *lenp, loff_t *ppos)
+ {
+ int r;
+ struct uts_namespace* uts_ns = current->nsproxy->uts_ns;
+ char* which;
+
+ /* map the filename to the pointer. perhaps it would be better
+ to put struct offset pointers in table->data ? */
+ switch (filp->f_dentry->d_name.name[3]) {
+ case 'y': /* ostype */
+ which = uts_ns->name.sysname;
+ break;
+ case 't': /* hostname */

```

```

+   which = uts_ns->name.nodename;
+   break;
+   case 'e': /* osrElease */
+   which = uts_ns->name.release;
+   break;
+   case 's': /* verSion */
+   which = uts_ns->name.version;
+   break;
+   case 'x': /* XXX - unreachable */
+   which = uts_ns->name.machine;
+   break;
+   case 'a': /* domAinname */
+   which = uts_ns->name.domainname;
+   break;
+   default:
+   printk("procfs: impossible uts part '%s'",
+         (char*)filp->f_dentry->d_name.name);
+   r = -EINVAL;
+   goto out;
+ }
+
+ if (!write) {
+   down_read(&uts_sem);
+   r=_proc_do_string(which,table->maxlen,0,filp,buffer,lenp, ppos);
+   up_read(&uts_sem);
+ } else {
+   down_write(&uts_sem);
+   r=_proc_do_string(which,table->maxlen,1,filp,buffer,lenp, ppos);
+   up_write(&uts_sem);
+ }
+ out:
+ return r;
+}
+#endif /* !CONFIG_UTS_NS */

```

```

static int do_proc_dointvec_conv(int *negp, unsigned long *lvalp,
    int *valp,

```

Subject: Re: [PATCH 3/3] proc: make UTS-related sysctls utsns aware
 Posted by [Andrew Morton](#) on Wed, 24 May 2006 15:54:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

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

```

>
> Add a new function proc_do_utsns_string, that derives the pointer
> into the uts_namespace->name structure, currently based on the
> filename of the dentry in /proc, and calls _proc_do_string()

```

> ---
> RFC only - not tested yet. builds, though

So... is it tested yet?

You owe me three Signed-off-by:s, please.

Subject: Re: [PATCH 3/3] proc: make UTS-related sysctls utsns aware
Posted by [Dave Hansen](#) on Wed, 24 May 2006 18:02:57 GMT
[View Forum Message](#) <> [Reply to Message](#)

On Tue, 2006-05-23 at 13:23 +1200, Sam Vilain wrote:

```
>
> + /* map the filename to the pointer. perhaps it would be
> better
> + to put struct offset pointers in table->data ? */
> + switch (filp->f_dentry->d_name.name[3]) {
> +     case 'y': /* ostYpe */
> +         which = uts_ns->name.sysname;
> +         break;
> +     case 't': /* hosTname */
> +         which = uts_ns->name.nodename;
> +         break;
> +     case 'e': /* osrElease */
> +         which = uts_ns->name.release;
> +         break;
> +     case 's': /* verSion */
> +         which = uts_ns->name.version;
> +         break;
> +     case 'x': /* XXX - unreachable */
> +         which = uts_ns->name.machine;
> +         break;
> +     case 'a': /* domAinname */
> +         which = uts_ns->name.domainname;
> +         break;
> +     default:
> +         printk("procfs: impossible uts part '%s'",
> +             (char*)filp->f_dentry->d_name.name);
> +         r = -EINVAL;
> +         goto out;
> + }
```

Why not just switch on the ->ctl_name from the table? Wouldn't that be easier?

-- Dave

Subject: Re: [PATCH 3/3] proc: make UTS-related sysctls utsns aware
Posted by [Sam Vilain](#) on Thu, 25 May 2006 03:51:54 GMT
[View Forum Message](#) <> [Reply to Message](#)

Andrew Morton wrote:

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

>

>>Add a new function proc_do_utsns_string, that derives the pointer

>> into the uts_namespace->name structure, currently based on the

>> filename of the dentry in /proc, and calls _proc_do_string()

>> ---

>> RFC only - not tested yet. builds, though

>

>

> So... is it tested yet?

No, I was waiting for feedback

(hackhackhack)

ok, now it's tested

```
root@ken:~# ./chuts /bin/bash
```

```
root@ken:~# uname -a
```

```
Linux ken 2.6.17-rc4-mm2-g2214c350 #8 PREEMPT Thu May 25 09:32:27 NZST  
2006 x86_64 GNU/Linux
```

```
root@ken:~# hostname bert
```

```
root@ken:~# uname -a
```

```
Linux bert 2.6.17-rc4-mm2-g2214c350 #8 PREEMPT Thu May 25 09:32:27 NZST  
2006 x86_64 GNU/Linux
```

```
root@ken:~# exit
```

```
root@ken:~# uname -a
```

```
Linux ken 2.6.17-rc4-mm2-g2214c350 #8 PREEMPT Thu May 25 09:32:27 NZST  
2006 x86_64 GNU/Linux
```

```
root@ken:~# cat /proc/sys/kernel/hostname
```

```
ken
```

```
root@ken:~# ./chuts /bin/sh -c "hostname bert && cat
```

```
/proc/sys/kernel/hostname"
```

```
bert
```

```
root@ken:~# cat /proc/sys/kernel/hostname
```

```
ken
```

```
root@ken:~# ./chuts /bin/sh -c "echo 'bob' > /proc/sys/kernel/hostname  
&& uname -n"
```

```
bob
```

```
root@ken:~# uname -n
```

```
ken
```

> You owe me three Signed-off-by:s, please.

They should be coming your way separately very shortly. I've revised the third one (namespaces-utsname-sysctl-hack-cleanup-2.patch), see my reply to Dave Hansen's e-mail. The revised version is the one I tested.

--

Sam Vilain, Catalyst IT (NZ) Ltd.

phone: +64 4 499 2267 cell: +64 21 55 40 50

DDI: +64 4 803 2342 PGP ID: 0x66B25843

Subject: Re: [PATCH 3/3] proc: make UTS-related sysctls utsns aware

Posted by [Sam Vilain](#) on Thu, 25 May 2006 04:01:44 GMT

[View Forum Message](#) <> [Reply to Message](#)

Dave Hansen wrote:

> On Tue, 2006-05-23 at 13:23 +1200, Sam Vilain wrote:

>

> >> + /* map the filename to the pointer. perhaps it would be

> >> better

> >> + to put struct offset pointers in table->data ? */

> >> + switch (filp->f_dentry->d_name.name[3]) {

> >> + case 'y': /* ostype */

> >> + which = uts_ns->name.sysname;

> >> + break;

>

>

> Why not just switch on the ->ctl_name from the table? Wouldn't that be

> easier?

Ha! Sure would.

Subject: proc: make UTS-related sysctls utsns aware

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

Add a new function `proc_do_utsns_string`, that derives the pointer into the `uts_namespace->name` structure, currently based on the filename of the dentry in `/proc`, and calls `_proc_do_string()`

Signed-off-by: Sam Vilain <sam.vilain@catalyst.net.nz>

Cc: Serge E. Hallyn <serue@us.ibm.com>

Cc: Kirill Korotaev <dev@openvz.org>

Cc: "Eric W. Biederman" <ebiederm@xmission.com>

Cc: Herbert Poetzl <herbert@13thfloor.at>

Cc: Andrey Savochkin <saw@sw.ru>

Cc: Andrew Morton <akpm@osdl.org>

kernel/sysctl.c | 97 ++++++

1 files changed, 97 insertions(+), 0 deletions(-)

diff --git a/kernel/sysctl.c b/kernel/sysctl.c

index cf053fc..5e6837e 100644

--- a/kernel/sysctl.c

+++ b/kernel/sysctl.c

@@ -141,8 +141,13 @@ #endif

static int parse_table(int __user *, int, void __user *, size_t __user *, void __user *, size_t, ctl_table *, void **);

+#ifndef CONFIG_UTS_NS

static int proc_do_uts_string(ctl_table *table, int write, struct file *filp, void __user *buffer, size_t *lenp, loff_t *ppos);

+#else

+static int proc_do_utsns_string(ctl_table *table, int write, struct file *filp,

+ void __user *buffer, size_t *lenp, loff_t *ppos);

+#endif

static ctl_table root_table[];

static struct ctl_table_header root_table_header =

@@ -238,6 +243,7 @@ #endif

};

static ctl_table kern_table[] = {

+#ifndef CONFIG_UTS_NS

{

.ctl_name = KERN_OSTYPE,

.procname = "ostype",

@@ -283,6 +289,54 @@ static ctl_table kern_table[] = {

.proc_handler = &proc_do_uts_string,

.strategy = &sysctl_string,

},

+#else /* !CONFIG_UTS_NS */

+ {

+ .ctl_name = KERN_OSTYPE,

+ .procname = "ostype",

+ .data = NULL,

+ /* could maybe use __NEW_UTS_LEN here? */

+ .maxlen = sizeof(current->nsproxy->uts_ns->name.sysname),

+ .mode = 0444,

+ .proc_handler = &proc_do_utsns_string,

+ .strategy = &sysctl_string,

+ },

+ {

+ .ctl_name = KERN_OSRELEASE,

+ .procname = "osrelease",

+ .data = NULL,

+ .maxlen = sizeof(current->nsproxy->uts_ns->name.release),

```

+ .mode = 0444,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_VERSION,
+ .procname = "version",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.version),
+ .mode = 0444,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_NODENAME,
+ .procname = "hostname",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.nodename),
+ .mode = 0644,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ {
+ .ctl_name = KERN_DOMAINNAME,
+ .procname = "domainname",
+ .data = NULL,
+ .maxlen = sizeof(current->nsproxy->uts_ns->name.domainname),
+ .mode = 0644,
+ .proc_handler = &proc_do_utsns_string,
+ .strategy = &sysctl_string,
+ },
+ #endif /* !CONFIG_UTS_NS */
+ {
+ .ctl_name = KERN_PANIC,
+ .procname = "panic",
@@ -1684,6 +1738,7 @@ int proc_dostiring(ctl_table *table, int
* to observe. Should this be in kernel/sys.c ???
*/

+ #ifndef CONFIG_UTS_NS
+ static int proc_do_uts_string(ctl_table *table, int write, struct file *filp,
+ void __user *buffer, size_t *lenp, loff_t *ppos)
+ {
@@ -1700,6 +1755,48 @@ static int proc_do_uts_string(ctl_table
}
return r;
}
+ #else /* !CONFIG_UTS_NS */

```

```

+static int proc_do_utsns_string(ctl_table *table, int write, struct file *filp,
+ void __user *buffer, size_t *lenp, loff_t *ppos)
+{
+ int r;
+ struct uts_namespace* uts_ns = current->nsproxy->uts_ns;
+ char* which;
+
+ switch (table->ctl_name) {
+ case KERN_OSTYPE:
+ which = uts_ns->name.sysname;
+ break;
+ case KERN_NODENAME:
+ which = uts_ns->name.nodename;
+ break;
+ case KERN_OSRELEASE:
+ which = uts_ns->name.release;
+ break;
+ case KERN_VERSION:
+ which = uts_ns->name.version;
+ break;
+ case KERN_DOMAINNAME:
+ which = uts_ns->name.domainname;
+ break;
+ default:
+ r = -EINVAL;
+ goto out;
+ }
+
+ if (!write) {
+ down_read(&uts_sem);
+ r=_proc_do_string(which,table->maxlen,0,filp,buffer,lenp, ppos);
+ up_read(&uts_sem);
+ } else {
+ down_write(&uts_sem);
+ r=_proc_do_string(which,table->maxlen,1,filp,buffer,lenp, ppos);
+ up_write(&uts_sem);
+ }
+ out:
+ return r;
+}
+#endif /* !CONFIG_UTS_NS */

static int do_proc_dointvec_conv(int *negp, unsigned long *lvalp,
int *valp,

```
