Subject: [PATCH] namespaces: uts_ns: make information visible via /proc/PID/uts directory
Posted by Sam Vilain on Mon, 22 May 2006 05:24:25 GMT
View Forum Message <> Reply to Message

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

Export the UTS information to a per-process directory /proc/PID/uts,
that has individual nodes for hostname, ostype, etc - similar to
those in /proc/sys/kernel

This duplicates the approach used for /proc/PID/attr, which involves a
lot of duplication of similar functions. Much room for maintenance
optimisation of both implementations remains.

Sorry for the duplication of this to the list, stuffed up the stgit
command.

After doing this I noticed that the whole way this is done via sysctls
in /proc/sys is much, much nicer. I was going there to make
/proc/sys/kernel/osname -> /proc/self/uts/sysname (etc), but it seems
that symlinks from /proc/sys are not a done thing.

Is there an argument here perhaps for some integration between the way
this is done for /proc/sys and /proc/PID/xxx ?

```
 fs/proc/base.c |  236 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 1 files changed, 236 insertions(+), 0 deletions(-)

diff --git a/fs/proc/base.c b/fs/proc/base.c
index 2031913..76f5acb 100644
--- a/fs/proc/base.c
+++ b/fs/proc/base.c
@@ -73,6 +73,7 @@ #include <linux/cpuset.h>
 #include <linux/audit.h>
 #include <linux/poll.h>
 #include <linux/nsproxy.h>
+#include <linux/utsname.h>
 #include "internal.h"

 /* NOTE:
@@ -179,6 +180,22 @@ #ifdef CONFIG_AUDITSYSCALL
 #endif
  PROC_TID_OOM_SCORE,
  PROC_TID_OOM_ADJUST,
+#ifdef CONFIG_UTS_NS
+ PROC_TID_UTS,
+ PROC_TGID_UTS,
+ PROC_TGID_UTS_SYSNAME,
+ PROC_TGID_UTS_NODENAME,
+ PROC_TGID_UTS_RELEASE,
+ PROC_TGID_UTS_VERSION,
+ PROC_TGID_UTS_MACHINE,
+ PROC_TGID_UTS_DOMAINNAME,
+ PROC_TID_UTS_SYSNAME,
+ PROC_TID_UTS_NODENAME,
+ PROC_TID_UTS_RELEASE,
+ PROC_TID_UTS_VERSION,
+ PROC_TID_UTS_MACHINE,
+ PROC_TID_UTS_DOMAINNAME,
+#endif

  /* Add new entries before this */
  PROC_TID_FD_DIR = 0x8000, /* 0x8000-0xffff */
@@ -238,6 +255,9 @@ #endif
 #ifdef CONFIG_AUDITSYSCALL
  E(PROC_TGID_LOGINUID, "loginuid", S_IFREG|S_IWUSR|S_IRUGO),
 #endif
+#ifdef CONFIG_UTS_NS
+ E(PROC_TGID_UTS,       "uts",    S_IFDIR|S_IRUGO|S_IXUGO),
+#endif
  {0,0,NULL,0}
 };
 static struct pid_entry tid_base_stuff[] = {
@@ -280,6 +300,9 @@ #endif
 #ifdef CONFIG_AUDITSYSCALL
  E(PROC_TID_LOGINUID, "loginuid", S_IFREG|S_IWUSR|S_IRUGO),
 #endif
+#ifdef CONFIG_UTS_NS
+ E(PROC_TID_UTS,        "uts",     S_IFDIR|S_IRUGO|S_IXUGO),
+#endif
  {0,0,NULL,0}
 };

@@ -300,6 +323,27 @@ static struct pid_entry tid_attr_stuff[]
 };
 #endif

+#ifdef CONFIG_UTS_NS
+static struct pid_entry tgid_uts_stuff[] = {
+ E(PROC_TGID_UTS_SYSNAME,    "sysname",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TGID_UTS_NODENAME,   "nodename",   S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TGID_UTS_RELEASE,    "release",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TGID_UTS_VERSION,    "version",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TGID_UTS_MACHINE,    "machine",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TGID_UTS_DOMAINNAME, "domainname", S_IFREG|S_IRUGO|S_IWUGO),
+ {0,0,NULL,0}
+};
+static struct pid_entry tid_uts_stuff[] = {
+ E(PROC_TID_UTS_SYSNAME,    "sysname",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TID_UTS_NODENAME,   "nodename",   S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TID_UTS_RELEASE,    "release",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TID_UTS_VERSION,    "version",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TID_UTS_MACHINE,    "machine",    S_IFREG|S_IRUGO|S_IWUGO),
+ E(PROC_TID_UTS_DOMAINNAME, "domainname", S_IFREG|S_IRUGO|S_IWUGO),
+ {0,0,NULL,0}
+};
+#endif
+
 #undef E

 static int proc_fd_link(struct inode *inode, struct dentry **dentry, struct vfsmount **mnt)
@@ -1608,6 +1652,148 @@ static struct file_operations proc_tgid_
 static struct inode_operations proc_tgid_attr_inode_operations;
 #endif

+#ifdef CONFIG_UTS_NS
+static ssize_t proc_pid_uts_read(struct file * file, char __user * buf,
+      size_t count, loff_t *ppos)
+{
+ struct inode * inode = file->f_dentry->d_inode;
+ ssize_t length;
+ loff_t __ppos = *ppos;
+ struct task_struct *task = get_proc_task(inode);
+ char __buf[__NEW_UTS_LEN+1];
+ char *which;
+
+ length = -ESRCH;
+ if (!task)
+  goto out_no_task;
+
+ switch (file->f_dentry->d_name.name[0]) {
+  case 's':
+   which = task->nsproxy->uts_ns->name.sysname;
+   break;
+  case 'n':
+   which = task->nsproxy->uts_ns->name.nodename;
+   break;
+  case 'r':
+   which = task->nsproxy->uts_ns->name.release;
+   break;
+  case 'v':
+   which = task->nsproxy->uts_ns->name.version;
+   break;
+  case 'm':
+   which = task->nsproxy->uts_ns->name.machine;
+   break;
+  case 'd':
+   which = task->nsproxy->uts_ns->name.domainname;
+   break;
+  default:
+   printk("procfs: impossible uts part '%s'",
+          (char*)file->f_dentry->d_name.name);
+   length = -EINVAL;
+   goto out;
+ }
+
+ length = strlen(which);
+ strcpy(__buf, which);
+ __buf[length++] = '\n';
+
+ if (__ppos >= length)
+  return 0;
+ if (count > length - __ppos)
+  count = length - __ppos;
+ if (copy_to_user(buf, __buf + __ppos, count))
+  return -EFAULT;
+
+out:
+ put_task_struct(task);
+out_no_task:
+ return length;
+}
+
+static ssize_t proc_pid_uts_write(struct file * file, const char __user * buf,
+       size_t count, loff_t *ppos)
+{
+ struct inode * inode = file->f_dentry->d_inode;
+ ssize_t length;
+ struct task_struct *task = get_proc_task(inode);
+ char *which;
+ char __buf[__NEW_UTS_LEN+1];
+
+ length = -ESRCH;
+ if (!task)
+  goto out_no_task;
+ if (count > PAGE_SIZE)
+  count = PAGE_SIZE;
+
+ /* No partial writes. */
+ length = -EINVAL;
+ if (*ppos != 0)
+  goto out;
+ if (count > __NEW_UTS_LEN)
+  goto out;
+
+ length = -EPERM;
+ if (!capable(CAP_SYS_ADMIN))
+  goto out;
+
+ length = -EFAULT;
+ if (copy_from_user(__buf, buf, count))
+  goto out;
+
+ length = -EINVAL;
+ length = strnlen(__buf, count);
+ if (count != length)
+  goto out;
+
+ if (__buf[length-1] == '\n')
+  length = length - 1;
+ __buf[length] = '\0';
+
+ switch (file->f_dentry->d_name.name[0]) {
+  case 's':
+   which = task->nsproxy->uts_ns->name.sysname;
+   break;
+  case 'n':
+   which = task->nsproxy->uts_ns->name.nodename;
+   break;
+  case 'r':
+   which = task->nsproxy->uts_ns->name.release;
+   break;
+  case 'v':
+   which = task->nsproxy->uts_ns->name.version;
+   break;
+  case 'm':
+   which = task->nsproxy->uts_ns->name.machine;
+   break;
+  case 'd':
+   which = task->nsproxy->uts_ns->name.domainname;
+   break;
+  default:
+   printk("procfs: impossible uts part '%s'",
+          (char*)file->f_dentry->d_name.name);
+   length = -EINVAL;
+   goto out;
+ }
+
+ strcpy(which, __buf);
```

```
+
+out:
+ put_task_struct(task);
+out_no_task:
+ return length;
+}
+
+static struct file_operations proc_pid_uts_operations = {
+ .read  = proc_pid_uts_read,
+ .write  = proc_pid_uts_write,
+};
+
+static struct file_operations proc_tid_uts_operations;
+static struct inode_operations proc_tid_uts_inode_operations;
+static struct file_operations proc_tgid_uts_operations;
+static struct inode_operations proc_tgid_uts_inode_operations;
+#endif
+
 /* SMP-safe */
 static struct dentry *proc_pident_lookup(struct inode *dir,
       struct dentry *dentry,
@@ -1760,6 +1946,30 @@ #ifdef CONFIG_SECURITY
   case PROC_TGID_ATTR_FSCREATE:
   inode->i_fop = &proc_pid_attr_operations;
   break;
+  case PROC_TID_UTS:
+   inode->i_nlink = 2;
+   inode->i_op = &proc_tid_uts_inode_operations;
+   inode->i_fop = &proc_tid_uts_operations;
+   break;
+  case PROC_TGID_UTS:
+   inode->i_nlink = 2;
+   inode->i_op = &proc_tgid_uts_inode_operations;
+   inode->i_fop = &proc_tgid_uts_operations;
+   break;
+  case PROC_TGID_UTS_SYSNAME:
+  case PROC_TGID_UTS_NODENAME:
+  case PROC_TGID_UTS_RELEASE:
+  case PROC_TGID_UTS_VERSION:
+  case PROC_TGID_UTS_MACHINE:
+  case PROC_TGID_UTS_DOMAINNAME:
+  case PROC_TID_UTS_SYSNAME:
+  case PROC_TID_UTS_NODENAME:
+  case PROC_TID_UTS_RELEASE:
+  case PROC_TID_UTS_VERSION:
+  case PROC_TID_UTS_MACHINE:
+  case PROC_TID_UTS_DOMAINNAME:
+   inode->i_fop = &proc_pid_uts_operations;
+   break;
 #endif
 #ifdef CONFIG_KALLSYMS
   case PROC_TID_WCHAN:
@@ -1889,6 +2099,32 @@ static struct inode_operations proc_tid_
 };
 #endif

+#ifdef CONFIG_UTS_NS
+static int proc_tgid_uts_readdir(struct file * filp,
+        void * dirent, filldir_t filldir)
+{
+ return proc_pident_readdir(filp,dirent,filldir,
+       tgid_uts_stuff,ARRAY_SIZE(tgid_uts_stuff));
+}
+
+static int proc_tid_uts_readdir(struct file * filp,
+        void * dirent, filldir_t filldir)
+{
+ return proc_pident_readdir(filp,dirent,filldir,
+       tid_uts_stuff,ARRAY_SIZE(tid_uts_stuff));
+}
+
+static struct file_operations proc_tgid_uts_operations = {
+ .read  = generic_read_dir,
+ .readdir = proc_tgid_uts_readdir,
+};
+
+static struct file_operations proc_tid_uts_operations = {
+ .read  = generic_read_dir,
+ .readdir = proc_tid_uts_readdir,
+};
+#endif
+
 /*
  * /proc/self:
  */
```