Subject: [PATCH 0/7] uts namespaces: Introduction
Posted by serue on Sat, 08 Apr 2006 04:52:06 GMT
View Forum Message <> Reply to Message

Introduce utsname namespaces. Instead of a single system_utsname
containing hostname domainname etc, a process can request it's
copy of the uts info to be cloned. The data will be copied from
it's original, but any further changes will not be seen by processes
which are not it's children, and vice versa.

This is useful, for instance, for vserver/openvz, which can now clone
a new uts namespace for each new virtual server.

Aside from the debugging patch which comes last, this patchset does
not actually implement a way for processes to unshare the uts namespace.
The proper unsharing semantics are to be worked out later.

Changes since last submission:
Restructured patchset so it compiles after each patch
Removed EXPORT_SYMBOL for unshare_uts_ns and free_uts_ns.
The former is now in the debugging pach and the latter gone
entirely, as unsharing is likely not something to be done
from modules!

-serge

Subject: [PATCH 7/7] uts namespaces: enable UTS namespaces debugging
Posted by serue on Sat, 08 Apr 2006 04:52:07 GMT
View Forum Message <> Reply to Message

Adds a /uts directory in debugfs which exposes the current UTS
namespace. If a file in this directory is changed, it will
unshare and modify the UTS namespace of the current process.

Clearly this is purely for testing purposes. Testing namespaces
in this fashion allows us to postpone concensus on a namespace
unsharing mechanism.

Signed-off-by: Cedric Le Goater <clg@fr.ibm.com>
Signed-off-by: Serge Hallyn <serue@us.ibm.com>

---

fs/debugfs/Makefile | 2 -
fs/debugfs/uts.c | 170 ++
init/version.c | 1
lib/Kconfig.debug | 11 +++

```
 4 files changed, 183 insertions(+), 1 deletions(-)
 create mode 100644 fs/debugfs/uts.c

13a0daa07ae35fa1b00250a2d14a9dfa8e70999b
diff --git a/fs/debugfs/Makefile b/fs/debugfs/Makefile
index 840c456..e5df8b9 100644
--- a/fs/debugfs/Makefile
+++ b/fs/debugfs/Makefile
@@ -1,4 +1,4 @@
 debugfs-objs := inode.o file.o

 obj-$(CONFIG_DEBUG_FS) += debugfs.o
-
+obj-$(CONFIG_DEBUG_UTS_NS) += uts.o
diff --git a/fs/debugfs/uts.c b/fs/debugfs/uts.c
new file mode 100644
index 0000000..45b29af
--- /dev/null
+++ b/fs/debugfs/uts.c
@@ -0,0 +1,170 @@
+/*
+ *  uts.c - adds a uts/ directory to debug UTS namespaces
+ *
+ *  Copyright (C) 2006 IBM
+ *
+ *  Author: Cedric Le Goater <clg@fr.ibm.com>
+ *
+ * This program is free software; you can redistribute it and/or
+ * modify it under the terms of the GNU General Public License as
+ * published by the Free Software Foundation, version 2 of the
+ * License.
+ */
+
+#include <linux/module.h>
+#include <linux/kernel.h>
+#include <linux/pagemap.h>
+#include <linux/debugfs.h>
+#include <linux/utsname.h>
+
+static struct dentry *uts_dentry;
+static struct dentry *uts_dentry_sysname;
+static struct dentry *uts_dentry_nodename;
+static struct dentry *uts_dentry_release;
+static struct dentry *uts_dentry_version;
+static struct dentry *uts_dentry_machine;
+static struct dentry *uts_dentry_domainname;
+
+static inline char* uts_buffer(struct dentry *dentry)
```

```
+{
+ if (dentry == uts_dentry_sysname)
+  return utsname()->sysname;
+ else if (dentry == uts_dentry_nodename)
+  return utsname()->nodename;
+ else if (dentry == uts_dentry_release)
+  return utsname()->release;
+ else if (dentry == uts_dentry_version)
+  return utsname()->version;
+ else if (dentry == uts_dentry_machine)
+  return utsname()->machine;
+ else if (dentry == uts_dentry_domainname)
+  return utsname()->domainname;
+ else
+  return NULL;
+}
+
+static ssize_t uts_read_file(struct file *file, char __user *user_buf,
+        size_t count, loff_t *ppos)
+{
+ size_t len;
+ char* buf;
+
+ if (*ppos < 0)
+  return -EINVAL;
+ if (*ppos >= count)
+  return 0;
+
+ buf = uts_buffer(file->f_dentry);
+ if (!buf)
+  return -ENOENT;
+
+ len = strlen(buf);
+ if (len > count)
+  len = count;
+ if (len)
+  if (copy_to_user(user_buf, buf, len))
+   return -EFAULT;
+ if (len < count) {
+  if (put_user('\n', ((char __user *) user_buf) + len))
+   return -EFAULT;
+  len++;
+ }
+
+ *ppos += count;
+ return count;
+}
+
```

```
+
+static ssize_t uts_write_file(struct file * file, const char __user * user_buf,
+         size_t count, loff_t *ppos)
+
+{
+ size_t len;
+ const char __user *p;
+ char c;
+ char* buf;
+
+ if (!unshare_uts_ns())
+  return -ENOMEM;
+
+ buf = uts_buffer(file->f_dentry);
+ if (!buf)
+  return -ENOENT;
+
+ len = 0;
+ p = user_buf;
+ while (len < count) {
+  if (get_user(c, p++))
+   return -EFAULT;
+  if (c == 0 || c == '\n')
+   break;
+  len++;
+ }
+
+ if (len >= __NEW_UTS_LEN)
+  len = __NEW_UTS_LEN - 1;
+
+ if (copy_from_user(buf, user_buf, len))
+  return -EFAULT;
+
+ buf[len] = 0;
+
+ *ppos += count;
+ return count;
+}
+
+static int uts_open(struct inode *inode, struct file *file)
+{
+ return 0;
+}
+
+static struct file_operations uts_file_operations = {
+ .read =  uts_read_file,
+ .write = uts_write_file,
+ .open =  uts_open,
```

```
+};
+
+static int __init uts_init(void)
+{
+ uts_dentry = debugfs_create_dir("uts", NULL);
+
+ uts_dentry_sysname = debugfs_create_file("sysname", 0666,
+       uts_dentry, NULL,
+       &uts_file_operations);
+ uts_dentry_nodename = debugfs_create_file("nodename", 0666,
+        uts_dentry, NULL,
+        &uts_file_operations);
+ uts_dentry_release = debugfs_create_file("release", 0666,
+       uts_dentry, NULL,
+       &uts_file_operations);
+ uts_dentry_version = debugfs_create_file("version", 0666,
+       uts_dentry, NULL,
+       &uts_file_operations);
+ uts_dentry_machine = debugfs_create_file("machine", 0666,
+       uts_dentry, NULL,
+       &uts_file_operations);
+ uts_dentry_domainname = debugfs_create_file("domainname", 0666,
+          uts_dentry, NULL,
+          &uts_file_operations);
+ return 0;
+}
+
+static void __exit uts_exit(void)
+{
+ debugfs_remove(uts_dentry_sysname);
+ debugfs_remove(uts_dentry_nodename);
+ debugfs_remove(uts_dentry_release);
+ debugfs_remove(uts_dentry_version);
+ debugfs_remove(uts_dentry_machine);
+ debugfs_remove(uts_dentry_domainname);
+ debugfs_remove(uts_dentry);
+}
+
+module_init(uts_init);
+module_exit(uts_exit);
+
+
+MODULE_DESCRIPTION("UTS namespace debugfs");
+MODULE_AUTHOR("Cedric Le Goater <clg@fr.ibm.com>");
+MODULE_LICENSE("GPL");
+
diff --git a/init/version.c b/init/version.c
index c05d8f8..9c83cce 100644
```

```
--- a/init/version.c
+++ b/init/version.c
@@ -70,6 +70,7 @@ struct uts_namespace *unshare_uts_ns(voi
  }
  return new_ns;
 }
+EXPORT_SYMBOL_GPL(unshare_uts_ns);

 void free_uts_ns(struct kref *kref)
 {
diff --git a/lib/Kconfig.debug b/lib/Kconfig.debug
index d57fd91..5ed09b8 100644
--- a/lib/Kconfig.debug
+++ b/lib/Kconfig.debug
@@ -223,3 +223,14 @@ config RCU_TORTURE_TEST
    at boot time (you probably don't).
    Say M if you want the RCU torture tests to build as a module.
    Say N if you are unsure.
+
+config DEBUG_UTS_NS
+ tristate "UTS Namespaces debugging"
+ depends on UTS_NS
+ select DEBUG_FS
+ default n
+ help
+   This option provides a kernel module that adds a uts/ directory
+   in debugfs which can be used to unshare and modify the uts namespace
+          of the current process and children. If unsure, say N.
+
--
1.2.4
```

Subject: [PATCH 6/7] uts namespaces: remove system_utsname
Posted by serue on Sat, 08 Apr 2006 04:52:07 GMT

View Forum Message <> Reply to Message

The system_utsname isn't needed now that kernel/sysctl.c is fixed.
Nuke it.

Signed-off-by: Serge E. Hallyn <serue@us.ibm.com>

```
 include/linux/utsname.h |    2 --
 1 files changed, 0 insertions(+), 2 deletions(-)
```

56b0a8c5ec7ee4f6cb48de587aa7d83d96d1f64d

```
diff --git a/include/linux/utsname.h b/include/linux/utsname.h
index e6ce607..a3bfc83 100644
--- a/include/linux/utsname.h
+++ b/include/linux/utsname.h
@@ -73,7 +73,5 @@ static inline struct new_utsname *init_u
  return &init_uts_ns.name;
 }

-#define system_utsname init_uts_ns.name
-
 extern struct rw_semaphore uts_sem;
 #endif
--
1.2.4
```

Subject: Re: [PATCH 0/7] uts namespaces: Introduction
Posted by Sam Vilain on Mon, 10 Apr 2006 01:15:54 GMT
View Forum Message <> Reply to Message

Serge,

I have just imported your series into the GIT repository where I have been collating the various recent related submissions at:

  git://vserver.utsl.gen.nz/vserver

A summary of the submissions imported to date are at:

  http://www.utsl.gen.nz/gitweb/?p=vserver;a=heads

I will endeavour to continue to collect and catalogue all vserver related submissions I receive, see on LKML, or get pull requests for, as a part of my efforts to merge this functionality.

Sam.

Serge E. Hallyn wrote:

>Introduce utsname namespaces. Instead of a single system_utsname
>containing hostname domainname etc, a process can request it's
>copy of the uts info to be cloned. The data will be copied from
>it's original, but any further changes will not be seen by processes
>which are not it's children, and vice versa.
>
>This is useful, for instance, for vserver/openvz, which can now clone
>a new uts namespace for each new virtual server.
>

>Aside from the debugging patch which comes last, this patchset does
>not actually implement a way for processes to unshare the uts namespace.
>The proper unsharing semantics are to be worked out later.
>
>Changes since last submission:
> Restructured patchset so it compiles after each patch
> Removed EXPORT_SYMBOL for unshare_uts_ns and free_uts_ns.
>   The former is now in the debugging pach and the latter gone
>   entirely, as unsharing is likely not something to be done
>   from modules!
>
>-serge
>
>
>
>