Subject: [PATCH 05/10] Task Containers(V11): Add procfs interface
Posted by Paul Menage on Fri, 20 Jul 2007 18:31:57 GMT
View Forum Message <> Reply to Message

This patch adds:

/proc/containers - general system info

/proc/*/container - per-task container membership info

Signed-off-by: Paul Menage <menage@google.com>

```
 fs/proc/base.c           |    7 ++
 include/linux/container.h |    2
 kernel/container.c       |  132 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 3 files changed, 141 insertions(+)

Index: container-2.6.22-rc6-mm1/fs/proc/base.c
 ========================================================== =======
--- container-2.6.22-rc6-mm1.orig/fs/proc/base.c
+++ container-2.6.22-rc6-mm1/fs/proc/base.c
@@ -67,6 +67,7 @@
 #include <linux/mount.h>
 #include <linux/security.h>
 #include <linux/ptrace.h>
+#include <linux/container.h>
 #include <linux/cpuset.h>
 #include <linux/audit.h>
 #include <linux/poll.h>
@@ -2050,6 +2051,9 @@ static const struct pid_entry tgid_base_
 #ifdef CONFIG_CPUSETS
   REG("cpuset",     S_IRUGO, cpuset),
 #endif
+#ifdef CONFIG_CONTAINERS
+ REG("container",  S_IRUGO, container),
+#endif
   INF("oom_score",  S_IRUGO, oom_score),
   REG("oom_adj",    S_IRUGO|S_IWUSR, oom_adjust),
 #ifdef CONFIG_AUDITSYSCALL
@@ -2341,6 +2345,9 @@ static const struct pid_entry tid_base_s
 #ifdef CONFIG_CPUSETS
   REG("cpuset",    S_IRUGO, cpuset),
 #endif
+#ifdef CONFIG_CONTAINERS
+ REG("container",  S_IRUGO, container),
+#endif
   INF("oom_score", S_IRUGO, oom_score),
```

```
  REG("oom_adj",   S_IRUGO|S_IWUSR, oom_adjust),
 #ifdef CONFIG_AUDITSYSCALL
Index: container-2.6.22-rc6-mm1/kernel/container.c
 ===================================================================
--- container-2.6.22-rc6-mm1.orig/kernel/container.c
+++ container-2.6.22-rc6-mm1/kernel/container.c
@@ -33,6 +33,7 @@
 #include <linux/mutex.h>
 #include <linux/mount.h>
 #include <linux/pagemap.h>
+#include <linux/proc_fs.h>
 #include <linux/rcupdate.h>
 #include <linux/sched.h>
 #include <linux/seq_file.h>
@@ -247,6 +248,7 @@ static int container_mkdir(struct inode
 static int container_rmdir(struct inode *unused_dir, struct dentry *dentry);
 static int container_populate_dir(struct container *cont);
 static struct inode_operations container_dir_inode_operations;
+static struct file_operations proc_containerstats_operations;

 static struct inode *container_new_inode(mode_t mode, struct super_block *sb)
 {
@@ -1567,6 +1569,7 @@ int __init container_init(void)
 {
  int err;
  int i;
+ struct proc_dir_entry *entry;

  for (i = 0; i < CONTAINER_SUBSYS_COUNT; i++) {
   struct container_subsys *ss = subsys[i];
@@ -1578,10 +1581,139 @@ int __init container_init(void)
  if (err < 0)
   goto out;

+ entry = create_proc_entry("containers", 0, NULL);
+ if (entry)
+  entry->proc_fops = &proc_containerstats_operations;
+
 out:
  return err;
 }

+/*
+ * proc_container_show()
+ *  - Print task's container paths into seq_file, one line for each hierarchy
+ *  - Used for /proc/<pid>/container.
+ *  - No need to task_lock(tsk) on this tsk->container reference, as it
+ *    doesn't really matter if tsk->container changes after we read it,
```

```
+ *    and we take container_mutex, keeping attach_task() from changing it
+ *    anyway.  No need to check that tsk->container != NULL, thanks to
+ *    the_top_container_hack in container_exit(), which sets an exiting tasks
+ *    container to top_container.
+ */
+
+/* TODO: Use a proper seq_file iterator */
+static int proc_container_show(struct seq_file *m, void *v)
+{
+ struct pid *pid;
+ struct task_struct *tsk;
+ char *buf;
+ int retval;
+ struct containerfs_root *root;
+
+ retval = -ENOMEM;
+ buf = kmalloc(PAGE_SIZE, GFP_KERNEL);
+ if (!buf)
+  goto out;
+
+ retval = -ESRCH;
+ pid = m->private;
+ tsk = get_pid_task(pid, PIDTYPE_PID);
+ if (!tsk)
+  goto out_free;
+
+ retval = 0;
+
+ mutex_lock(&container_mutex);
+
+ for_each_root(root) {
+  struct container_subsys *ss;
+  struct container *cont;
+  int subsys_id;
+  int count = 0;
+
+  /* Skip this hierarchy if it has no active subsystems */
+  if (!root->actual_subsys_bits)
+   continue;
+  for_each_subsys(root, ss)
+   seq_printf(m, "%s%s", count++ ? "," : "", ss->name);
+  seq_putc(m, ':');
+  get_first_subsys(&root->top_container, NULL, &subsys_id);
+  cont = task_container(tsk, subsys_id);
+  retval = container_path(cont, buf, PAGE_SIZE);
+  if (retval < 0)
+   goto out_unlock;
+  seq_puts(m, buf);
```

```
+  seq_putc(m, '\n');
+ }
+
+out_unlock:
+ mutex_unlock(&container_mutex);
+ put_task_struct(tsk);
+out_free:
+ kfree(buf);
+out:
+ return retval;
+}
+
+static int container_open(struct inode *inode, struct file *file)
+{
+ struct pid *pid = PROC_I(inode)->pid;
+ return single_open(file, proc_container_show, pid);
+}
+
+struct file_operations proc_container_operations = {
+ .open  = container_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+/* Display information about each subsystem and each hierarchy */
+static int proc_containerstats_show(struct seq_file *m, void *v)
+{
+ int i;
+ struct containerfs_root *root;
+
+ mutex_lock(&container_mutex);
+ seq_puts(m, "Hierarchies:\n");
+ for_each_root(root) {
+  struct container_subsys *ss;
+  int first = 1;
+  seq_printf(m, "%p: bits=%lx containers=%d (", root,
+      root->subsys_bits, root->number_of_containers);
+  for_each_subsys(root, ss) {
+   seq_printf(m, "%s%s", first ? "" : ", ", ss->name);
+   first = false;
+  }
+  seq_putc(m, ')');
+  if (root->sb) {
+   seq_printf(m, " s_active=%d",
+       atomic_read(&root->sb->s_active));
+  }
+  seq_putc(m, '\n');
```

```
+ }
+ seq_puts(m, "Subsystems:\n");
+ for (i = 0; i < CONTAINER_SUBSYS_COUNT; i++) {
+  struct container_subsys *ss = subsys[i];
+  seq_printf(m, "%d: name=%s hierarchy=%p\n",
+      i, ss->name, ss->root);
+ }
+ mutex_unlock(&container_mutex);
+ return 0;
+}
+
+static int containerstats_open(struct inode *inode, struct file *file)
+{
+ return single_open(file, proc_containerstats_show, 0);
+}
+
+static struct file_operations proc_containerstats_operations = {
+ .open = containerstats_open,
+ .read = seq_read,
+ .llseek = seq_lseek,
+ .release = single_release,
+};
+
 /**
  * container_fork - attach newly forked task to its parents container.
  * @tsk: pointer to task_struct of forking parent process.
Index: container-2.6.22-rc6-mm1/include/linux/container.h
 ===================================================================== =======
--- container-2.6.22-rc6-mm1.orig/include/linux/container.h
+++ container-2.6.22-rc6-mm1/include/linux/container.h
@@ -29,6 +29,8 @@ extern void container_fork(struct task_s
 extern void container_fork_callbacks(struct task_struct *p);
 extern void container_exit(struct task_struct *p, int run_callbacks);

+extern struct file_operations proc_container_operations;
+
 /* Per-subsystem/per-container state maintained by the system. */
 struct container_subsys_state {
  /* The container that this subsystem is attached to. Useful

--
```