Subject: [PATCH 06/33] task containersv11 add procfs interface
Posted by Paul Menage on Mon, 17 Sep 2007 21:03:13 GMT
View Forum Message <> Reply to Message

Add:

/proc/cgroups - general system info

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

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

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

diff -puN fs/proc/base.c~task-cgroupsv11-add-procfs-interface fs/proc/base.c
--- a/fs/proc/base.c~task-cgroupsv11-add-procfs-interface
+++ a/fs/proc/base.c
@@ -67,6 +67,7 @@
 #include <linux/mount.h>
 #include <linux/security.h>
 #include <linux/ptrace.h>
+#include <linux/cgroup.h>
 #include <linux/cpuset.h>
 #include <linux/audit.h>
 #include <linux/poll.h>
@@ -2051,6 +2052,9 @@ static const struct pid_entry tgid_base_
 #ifdef CONFIG_CPUSETS
   REG("cpuset",     S_IRUGO, cpuset),
 #endif
+#ifdef CONFIG_CGROUPS
+ REG("cgroup",  S_IRUGO, cgroup),
+#endif
   INF("oom_score",  S_IRUGO, oom_score),
   REG("oom_adj",    S_IRUGO|S_IWUSR, oom_adjust),
 #ifdef CONFIG_AUDITSYSCALL
@@ -2340,6 +2344,9 @@ static const struct pid_entry tid_base_s
 #ifdef CONFIG_CPUSETS
   REG("cpuset",    S_IRUGO, cpuset),
 #endif
+#ifdef CONFIG_CGROUPS
+ REG("cgroup",  S_IRUGO, cgroup),
+#endif
   INF("oom_score", S_IRUGO, oom_score),
```

```
  REG("oom_adj",   S_IRUGO|S_IWUSR, oom_adjust),
 #ifdef CONFIG_AUDITSYSCALL
diff -puN include/linux/cgroup.h~task-cgroupsv11-add-procfs-interface include/linux/cgroup.h
--- a/include/linux/cgroup.h~task-cgroupsv11-add-procfs-interface
+++ a/include/linux/cgroup.h
@@ -29,6 +29,8 @@ extern void cgroup_fork(struct task_s
 extern void cgroup_fork_callbacks(struct task_struct *p);
 extern void cgroup_exit(struct task_struct *p, int run_callbacks);

+extern struct file_operations proc_cgroup_operations;
+
 /* Per-subsystem/per-cgroup state maintained by the system. */
 struct cgroup_subsys_state {
  /* The cgroup that this subsystem is attached to. Useful
diff -puN kernel/cgroup.c~task-cgroupsv11-add-procfs-interface kernel/cgroup.c
--- a/kernel/cgroup.c~task-cgroupsv11-add-procfs-interface
+++ a/kernel/cgroup.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 cgroup_mkdir(struct inode
 static int cgroup_rmdir(struct inode *unused_dir, struct dentry *dentry);
 static int cgroup_populate_dir(struct cgroup *cont);
 static struct inode_operations cgroup_dir_inode_operations;
+static struct file_operations proc_cgroupstats_operations;

 static struct inode *cgroup_new_inode(mode_t mode, struct super_block *sb)
 {
@@ -1576,6 +1578,7 @@ int __init cgroup_init(void)
 {
  int err;
  int i;
+ struct proc_dir_entry *entry;

  for (i = 0; i < CGROUP_SUBSYS_COUNT; i++) {
   struct cgroup_subsys *ss = subsys[i];
@@ -1587,10 +1590,139 @@ int __init cgroup_init(void)
  if (err < 0)
   goto out;

+ entry = create_proc_entry("cgroups", 0, NULL);
+ if (entry)
+  entry->proc_fops = &proc_cgroupstats_operations;
```

```
+
 out:
  return err;
 }

+/*
+ * proc_cgroup_show()
+ *  - Print task's cgroup paths into seq_file, one line for each hierarchy
+ *  - Used for /proc/<pid>/cgroup.
+ *  - No need to task_lock(tsk) on this tsk->cgroup reference, as it
+ *    doesn't really matter if tsk->cgroup changes after we read it,
+ *    and we take cgroup_mutex, keeping attach_task() from changing it
+ *    anyway.  No need to check that tsk->cgroup != NULL, thanks to
+ *    the_top_cgroup_hack in cgroup_exit(), which sets an exiting tasks
+ *    cgroup to top_cgroup.
+ */
+
+/* TODO: Use a proper seq_file iterator */
+static int proc_cgroup_show(struct seq_file *m, void *v)
+{
+ struct pid *pid;
+ struct task_struct *tsk;
+ char *buf;
+ int retval;
+ struct cgroupfs_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(&cgroup_mutex);
+
+ for_each_root(root) {
+  struct cgroup_subsys *ss;
+  struct cgroup *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_cgroup, NULL, &subsys_id);
+  cont = task_cgroup(tsk, subsys_id);
+  retval = cgroup_path(cont, buf, PAGE_SIZE);
+  if (retval < 0)
+   goto out_unlock;
+  seq_puts(m, buf);
+  seq_putc(m, '\n');
+ }
+
+out_unlock:
+ mutex_unlock(&cgroup_mutex);
+ put_task_struct(tsk);
+out_free:
+ kfree(buf);
+out:
+ return retval;
+}
+
+static int cgroup_open(struct inode *inode, struct file *file)
+{
+ struct pid *pid = PROC_I(inode)->pid;
+ return single_open(file, proc_cgroup_show, pid);
+}
+
+struct file_operations proc_cgroup_operations = {
+ .open  = cgroup_open,
+ .read  = seq_read,
+ .llseek  = seq_lseek,
+ .release = single_release,
+};
+
+/* Display information about each subsystem and each hierarchy */
+static int proc_cgroupstats_show(struct seq_file *m, void *v)
+{
+ int i;
+ struct cgroupfs_root *root;
+
+ mutex_lock(&cgroup_mutex);
+ seq_puts(m, "Hierarchies:\n");
+ for_each_root(root) {
+  struct cgroup_subsys *ss;
+  int first = 1;
+  seq_printf(m, "%p: bits=%lx cgroups=%d (", root,
```

```
+      root->subsys_bits, root->number_of_cgroups);
+  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 < CGROUP_SUBSYS_COUNT; i++) {
+  struct cgroup_subsys *ss = subsys[i];
+  seq_printf(m, "%d: name=%s hierarchy=%p\n",
+      i, ss->name, ss->root);
+ }
+ mutex_unlock(&cgroup_mutex);
+ return 0;
+}
+
+static int cgroupstats_open(struct inode *inode, struct file *file)
+{
+ return single_open(file, proc_cgroupstats_show, 0);
+}
+
+static struct file_operations proc_cgroupstats_operations = {
+ .open = cgroupstats_open,
+ .read = seq_read,
+ .llseek = seq_lseek,
+ .release = single_release,
+};
+
 /**
  * cgroup_fork - attach newly forked task to its parents cgroup.
  * @tsk: pointer to task_struct of forking parent process.
_

--

_____
Containers mailing list
Containers@lists.linux-foundation.org
https://lists.linux-foundation.org/mailman/listinfo/containers
```