Subject: [RFC][PATCH 5/5] Add a Signal Control Group Subsystem
Posted by Matt Helsley on Thu, 24 Apr 2008 06:48:01 GMT
View Forum Message <> Reply to Message

Add a signal control group subsystem that allows us to send signals to all tasks
in the control group by writing the desired signal(7) number to the kill file.

NOTE: We don't really need per-cgroup state, but control groups doesn't support
stateless subsystems yet.

Signed-off-by: Matt Helsley <matthltc@us.ibm.com>
---
 include/linux/cgroup_signal.h | 28 +++++++++
 include/linux/cgroup_subsys.h | 6 +
 init/Kconfig | 6 +
 kernel/Makefile | 1
 kernel/cgroup_signal.c | 129 ++
 5 files changed, 170 insertions(+)

Index: linux-2.6.25-mm1/include/linux/cgroup_signal.h
===
--- /dev/null
+++ linux-2.6.25-mm1/include/linux/cgroup_signal.h
@@ -0,0 +1,28 @@
+#ifndef _LINUX_CGROUP_SIGNAL_H
+#define _LINUX_CGROUP_SIGNAL_H
+/*
+ * cgroup_signal.h - control group freezer subsystem interface
+ *
+ * Copyright IBM Corp. 2007
+ *
+ * Author : Cedric Le Goater <clg@fr.ibm.com>
+ * Author : Matt Helsley <matthltc@us.ibm.com>
+ */
+
+#include <linux/cgroup.h>
+
+#ifdef CONFIG_CGROUP_SIGNAL
+
+struct stateless {
+ struct cgroup_subsys_state css;
+};
+
+static inline struct stateless *cgroup_signal(struct cgroup *cgroup)
+{
+ return container_of(cgroup_subsys_state(cgroup, signal_subsys_id),
+   struct stateless, css);
+}

```
+
+#else /* !CONFIG_CGROUP_SIGNAL */
+#endif /* !CONFIG_CGROUP_SIGNAL */
+#endif /* _LINUX_CGROUP_SIGNAL_H */
Index: linux-2.6.25-mm1/kernel/cgroup_signal.c
===================================================================
--- /dev/null
+++ linux-2.6.25-mm1/kernel/cgroup_signal.c
@@ -0,0 +1,129 @@
+/*
+ * cgroup_signal.c -  control group signal subsystem
+ *
+ * Copyright IBM Corp. 2007
+ *
+ * Author : Cedric Le Goater <clg@fr.ibm.com>
+ * Author : Matt Helsley <matthltc@us.ibm.com>
+ */
+
+#include <linux/module.h>
+#include <linux/cgroup.h>
+#include <linux/fs.h>
+#include <linux/uaccess.h>
+#include <linux/cgroup_signal.h>
+
+struct cgroup_subsys signal_subsys;
+
+static struct cgroup_subsys_state *signal_create(
+ struct cgroup_subsys *ss, struct cgroup *cgroup)
+{
+ struct stateless *dummy;
+
+ if (!capable(CAP_SYS_ADMIN))
+  return ERR_PTR(-EPERM);
+
+ dummy = kzalloc(sizeof(struct stateless), GFP_KERNEL);
+ if (!dummy)
+  return ERR_PTR(-ENOMEM);
+ return  &dummy->css;
+}
+
+static void signal_destroy(struct cgroup_subsys *ss,
+       struct cgroup *cgroup)
+{
+ kfree(cgroup_signal(cgroup));
+}
+
+
+static int signal_can_attach(struct cgroup_subsys *ss,
```

```
+       struct cgroup *new_cgroup,
+       struct task_struct *task)
+{
+ return 0;
+}
+
+static int signal_kill(struct cgroup *cgroup, int signum)
+{
+ struct cgroup_iter it;
+ struct task_struct *task;
+ int retval = 0;
+
+ cgroup_iter_start(cgroup, &it);
+ while ((task = cgroup_iter_next(cgroup, &it))) {
+  retval = send_sig(signum, task, 1);
+  if (retval)
+   break;
+ }
+ cgroup_iter_end(cgroup, &it);
+
+ return retval;
+}
+
+static ssize_t signal_write(struct cgroup *cgroup,
+       struct cftype *cft,
+       struct file *file,
+       const char __user *userbuf,
+       size_t nbytes, loff_t *unused_ppos)
+{
+ char *buffer;
+ int retval = 0;
+ int value;
+
+ if (nbytes >= PATH_MAX)
+  return -E2BIG;
+
+ /* +1 for nul-terminator */
+ buffer = kmalloc(nbytes + 1, GFP_KERNEL);
+ if (buffer == NULL)
+  return -ENOMEM;
+
+ if (copy_from_user(buffer, userbuf, nbytes)) {
+  retval = -EFAULT;
+  goto free_buffer;
+ }
+ buffer[nbytes] = 0; /* nul-terminate */
+ if (sscanf(buffer, "%d", &value) != 1) {
+  retval = -EIO;
```

```
+  goto free_buffer;
+ }
+
+ cgroup_lock();
+
+ if (cgroup_is_removed(cgroup)) {
+  retval = -ENODEV;
+  goto unlock;
+ }
+
+ retval = signal_kill(cgroup, value);
+ if (retval == 0)
+  retval = nbytes;
+unlock:
+ cgroup_unlock();
+free_buffer:
+ kfree(buffer);
+ return retval;
+}
+
+static struct cftype kill_file = {
+ .name = "kill",
+ .write = signal_write,
+ .private = 0,
+};
+
+static int signal_populate(struct cgroup_subsys *ss, struct cgroup *cgroup)
+{
+ return cgroup_add_files(cgroup, ss, &kill_file, 1);
+}
+
+struct cgroup_subsys signal_subsys = {
+ .name  = "signal",
+ .create  = signal_create,
+ .destroy = signal_destroy,
+ .populate = signal_populate,
+ .subsys_id = signal_subsys_id,
+ .can_attach = signal_can_attach,
+ .attach  = NULL,
+ .fork  = NULL,
+ .exit  = NULL,
+};
Index: linux-2.6.25-mm1/init/Kconfig
===================================================================
--- linux-2.6.25-mm1.orig/init/Kconfig
+++ linux-2.6.25-mm1/init/Kconfig
@@ -328,10 +328,16 @@ config CGROUP_FREEZER
        depends on CGROUPS
```

```
        help
          Provides a way to freeze and unfreeze all tasks in a
    cgroup

+config CGROUP_SIGNAL
+       bool "control group signal subsystem"
+       depends on CGROUPS
+       help
+         Provides a way to signal all tasks in a cgroup
+
 config FAIR_GROUP_SCHED
  bool "Group scheduling for SCHED_OTHER"
  depends on GROUP_SCHED
  default y

Index: linux-2.6.25-mm1/kernel/Makefile
===================================================================
--- linux-2.6.25-mm1.orig/kernel/Makefile
+++ linux-2.6.25-mm1/kernel/Makefile
@@ -47,10 +47,11 @@ obj-$(CONFIG_KEXEC) += kexec.o
 obj-$(CONFIG_BACKTRACE_SELF_TEST) += backtracetest.o
 obj-$(CONFIG_COMPAT) += compat.o
 obj-$(CONFIG_CGROUPS) += cgroup.o
 obj-$(CONFIG_CGROUP_DEBUG) += cgroup_debug.o
 obj-$(CONFIG_CGROUP_FREEZER) += cgroup_freezer.o
+obj-$(CONFIG_CGROUP_SIGNAL) += cgroup_signal.o
 obj-$(CONFIG_CPUSETS) += cpuset.o
 obj-$(CONFIG_CGROUP_NS) += ns_cgroup.o
 obj-$(CONFIG_UTS_NS) += utsname.o
 obj-$(CONFIG_USER_NS) += user_namespace.o
 obj-$(CONFIG_PID_NS) += pid_namespace.o
Index: linux-2.6.25-mm1/include/linux/cgroup_subsys.h
===================================================================
--- linux-2.6.25-mm1.orig/include/linux/cgroup_subsys.h
+++ linux-2.6.25-mm1/include/linux/cgroup_subsys.h
@@ -52,5 +52,11 @@ SUBSYS(devices)
 #ifdef CONFIG_CGROUP_FREEZER
 SUBSYS(freezer)
 #endif

 /* */
+
+#ifdef CONFIG_CGROUP_SIGNAL
+SUBSYS(signal)
+#endif
+
+/* */
```

--

_____

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