
Subject: [RFC/PATCH 2/8]: CGroup Files: Add a cgroup write_string control file method

Posted by [Paul Menage](#) on Tue, 13 May 2008 06:37:09 GMT

[View Forum Message](#) <> [Reply to Message](#)

This patch adds a write_string() method for cgroups control files. The semantics are that a buffer is copied from userspace to kernelspace and the handler function invoked on that buffer. Any control group locking is done after the copy from userspace has occurred. The buffer is guaranteed to be nul-terminated, and no longer than max_write_len (defaulting to 64 bytes if unspecified). Later patches will convert existing raw file write handlers in control group subsystems to use this method.

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

include/linux/cgroup.h | 10 +++++++
kernel/cgroup.c | 5 +++-
2 files changed, 14 insertions(+), 1 deletion(-)

Index: cgroup-2.6.25-mm1/include/linux/cgroup.h

=====

--- cgroup-2.6.25-mm1.orig/include/linux/cgroup.h
+++ cgroup-2.6.25-mm1/include/linux/cgroup.h
@@ -281,6 +281,10 @@ struct cftype {
 */
 int lockmode;

+ /* If non-zero, defines the maximum length of string that can
+ * be passed to write_string; defaults to 64 */
+ int max_write_len;
+
 int (*open) (struct inode *inode, struct file *file);
 ssize_t (*read) (struct cgroup *cgrp, struct cftype *cft,
 struct file *file,
@@ -323,6 +327,12 @@ struct cftype {
 * write_s64() is a signed version of write_u64()
 */
 int (*write_s64) (struct cgroup *cgrp, struct cftype *cft, s64 val);
+ /*
+ * write_string() is passed a nul-terminated kernelspace
+ * buffer of maximum length determined by max_write_len
+ */
+ int (*write_string) (struct cgroup *cgrp, struct cftype *cft,
+ char *buffer);

 /*

* trigger() callback can be used to get some kick from the

Index: cgroup-2.6.25-mm1/kernel/cgroup.c

```
=====
--- cgroup-2.6.25-mm1.orig/kernel/cgroup.c
+++ cgroup-2.6.25-mm1/kernel/cgroup.c
@@ -1461,7 +1461,7 @@ static ssize_t cgroup_file_write(struct
     ssize_t retval;
     char static_buffer[64];
     char *buffer = static_buffer;
-    ssize_t max_bytes = sizeof(static_buffer) - 1;
+    ssize_t max_bytes = cft->max_write_len ? : sizeof(static_buffer) - 1;
     if (!cft->write && !cft->trigger) {
         if (!nbytes)
             return -EINVAL;
@@ -1489,6 +1489,8 @@ static ssize_t cgroup_file_write(struct
     retval = cft->write(cgrp, cft, file, userbuf, nbytes, ppos);
     else if (cft->write_u64 || cft->write_s64)
         retval = cgroup_write_X64(cgrp, cft, buffer);
+    else if (cft->write_string)
+        retval = cft->write_string(cgrp, cft, buffer);
     else if (cft->trigger)
         retval = cft->trigger(cgrp, (unsigned int)cft->private);
     else
@@ -1651,6 +1653,7 @@ static struct file_operations cgroup_seq
     .read = seq_read,
     .lseek = seq_lseek,
     .release = cgroup_seqfile_release,
+    .write = cgroup_file_write,
 };

 static int cgroup_file_open(struct inode *inode, struct file *file)

--
```

Containers mailing list

Containers@lists.linux-foundation.org

<https://lists.linux-foundation.org/mailman/listinfo/containers>
