Subject: [RFC/PATCH 7/8]: CGroup Files: Convert devcgroup_access_write() into a cgroup write_string() handler
Posted by Paul Menage on Tue, 13 May 2008 06:37:14 GMT
View Forum Message <> Reply to Message

This patch converts devcgroup_access_write() from a raw file handler
into a handler for the cgroup write_string() method. This allows some
boilerplate copying/locking/checking to be removed and simplifies the
cleanup path, since these functions are performed by the cgroups
framework before calling the handler.

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

---
 security/device_cgroup.c | 79 ++++++++++++++++++----------------------------
 1 file changed, 22 insertions(+), 57 deletions(-)

Index: cgroup-2.6.25-mm1/security/device_cgroup.c
===
--- cgroup-2.6.25-mm1.orig/security/device_cgroup.c
+++ cgroup-2.6.25-mm1/security/device_cgroup.c
@@ -323,14 +323,14 @@ static int parent_has_perm(struct cgroup
  * new access is only allowed if you're in the top-level cgroup, or your
  * parent cgroup has the access you're asking for.
  */
-static ssize_t devcgroup_access_write(struct cgroup *cgroup, struct cftype *cft,
-    struct file *file, const char __user *userbuf,
-    size_t nbytes, loff_t *ppos)
+static int devcgroup_access_write(struct cgroup *cgroup,
+   struct cftype *cft,
+   char *buffer)
 {
  struct cgroup *cur_cgroup;
  struct dev_cgroup *devcgroup, *cur_devcgroup;
  int filetype = cft->private;
- char *buffer, *b;
+ char *b;
  int retval = 0, count;
  struct dev_whitelist_item wh;

@@ -341,22 +341,6 @@ static ssize_t devcgroup_access_write(st
  cur_cgroup = task_cgroup(current, devices_subsys.subsys_id);
  cur_devcgroup = cgroup_to_devcgroup(cur_cgroup);

- buffer = kmalloc(nbytes+1, GFP_KERNEL);
- if (!buffer)
- return -ENOMEM;
-

```
- if (copy_from_user(buffer, userbuf, nbytes)) {
-  retval = -EFAULT;
-  goto out1;
- }
- buffer[nbytes] = 0; /* nul-terminate */
-
- cgroup_lock();
- if (cgroup_is_removed(cgroup)) {
-  retval = -ENODEV;
-  goto out2;
- }
-
  memset(&wh, 0, sizeof(wh));
  b = buffer;

@@ -372,14 +356,11 @@ static ssize_t devcgroup_access_write(st
   wh.type = DEV_CHAR;
   break;
  default:
-  retval = -EINVAL;
-  goto out2;
+  return -EINVAL;
  }
  b++;
- if (!isspace(*b)) {
-  retval = -EINVAL;
-  goto out2;
- }
+ if (!isspace(*b))
+  return -EINVAL;
  b++;
  if (*b == '*') {
   wh.major = ~0;
@@ -391,13 +372,10 @@ static ssize_t devcgroup_access_write(st
    b++;
   }
  } else {
-  retval = -EINVAL;
-  goto out2;
- }
- if (*b != ':') {
-  retval = -EINVAL;
-  goto out2;
+  return -EINVAL;
  }
+ if (*b != ':')
+  return -EINVAL;
  b++;
```

```
  /* read minor */
@@ -411,13 +389,10 @@ static ssize_t devcgroup_access_write(st
     b++;
    }
   } else {
-  retval = -EINVAL;
-  goto out2;
- }
- if (!isspace(*b)) {
-  retval = -EINVAL;
-  goto out2;
+  return -EINVAL;
   }
+ if (!isspace(*b))
+  return -EINVAL;
   for (b++, count = 0; count < 3; count++, b++) {
    switch (*b) {
    case 'r':
@@ -434,8 +409,7 @@ static ssize_t devcgroup_access_write(st
     count = 3;
     break;
    default:
-   retval = -EINVAL;
-   goto out2;
+   return -EINVAL;
    }
   }

@@ -444,38 +418,29 @@ handle:
   switch (filetype) {
   case DEVCG_ALLOW:
    if (!parent_has_perm(cgroup, &wh))
-   retval = -EPERM;
-  else
-   retval = dev_whitelist_add(devcgroup, &wh);
-  break;
+   return -EPERM;
+  return dev_whitelist_add(devcgroup, &wh);
   case DEVCG_DENY:
    dev_whitelist_rm(devcgroup, &wh);
    break;
   default:
-  retval = -EINVAL;
-  goto out2;
+  return -EINVAL;
   }
-
```

```
- if (retval == 0)
-  retval = nbytes;
-
-out2:
- cgroup_unlock();
-out1:
- kfree(buffer);
- return retval;
+ return 0;
 }

 static struct cftype dev_cgroup_files[] = {
  {
   .name = "allow",
-  .write  = devcgroup_access_write,
+  .write_string  = devcgroup_access_write,
   .private = DEVCG_ALLOW,
+  .lockmode = CFT_LOCK_CGL_WRITE,
  },
  {
   .name = "deny",
-  .write = devcgroup_access_write,
+  .write_string = devcgroup_access_write,
   .private = DEVCG_DENY,
+  .lockmode = CFT_LOCK_CGL_WRITE,
  },
  {
   .name = "list",

--
```