Subject: [PATCH 2/4] The character devices layer changes
Posted by Pavel Emelianov on Thu, 07 Feb 2008 12:59:04 GMT
View Forum Message <> Reply to Message

These changes include the API for the control group
to map/remap/unmap the devices with their permissions
and one important thing.

The fact is that the struct cdev is cached in the inode
for faster access, so once we looked one up we go through
the fast path and omit the kobj_lookup() call. This is no
longer good when we restrict the access to cdevs. Another
problem is that different char devices may use one struct
cdev object, so having an access to one of them grants us
access to another, so we have to re-lookup the kobj map
each open.

Signed-off-by: Pavel Emelyanov <xemul@openvz.org>

```
diff --git a/fs/char_dev.c b/fs/char_dev.c
index 2c7a8b5..ddacab7 100644
--- a/fs/char_dev.c
+++ b/fs/char_dev.c
@@ -22,6 +22,8 @@
 #include <linux/mutex.h>
 #include <linux/backing-dev.h>

+#include <linux/devscontrol.h>
+
 #ifdef CONFIG_KMOD
 #include <linux/kmod.h>
 #endif
@@ -362,17 +364,31 @@ int chrdev_open(struct inode * inode, struct file * filp)
  struct cdev *p;
  struct cdev *new = NULL;
  int ret = 0;
+ struct kobj_map *map;
+
+ map = task_cdev_map(current);
+ if (map == NULL)
+  map = cdev_map;

  spin_lock(&cdev_lock);
  p = inode->i_cdev;
- if (!p) {
+ if (!p || map != cdev_map) {
```

```
   struct kobject *kobj;
   int idx;
+  mode_t mode;
+
   spin_unlock(&cdev_lock);
-  kobj = kobj_lookup(cdev_map, inode->i_rdev, &idx);
+  kobj = kobj_lookup(map, inode->i_rdev, &mode, &idx);
   if (!kobj)
    return -ENXIO;
   new = container_of(kobj, struct cdev, kobj);
+  BUG_ON(p != NULL && p != new);
+
+  if ((filp->f_mode & mode) != filp->f_mode) {
+   cdev_put(new);
+   return -EACCES;
+  }
+
   spin_lock(&cdev_lock);
   p = inode->i_cdev;
   if (!p) {
@@ -461,6 +477,53 @@ int cdev_add(struct cdev *p, dev_t dev, unsigned count)
  return kobj_map(cdev_map, dev, count, NULL, exact_match, exact_lock, p);
 }

+#ifdef CONFIG_CGROUP_DEVS
+int cdev_add_to_map(struct kobj_map *map, dev_t dev, int all, mode_t mode)
+{
+ int tmp;
+ struct kobject *k;
+ struct cdev *c;
+
+ k = kobj_lookup(cdev_map, dev, NULL, &tmp);
+ if (k == NULL)
+  return -ENODEV;
+
+ c = container_of(k, struct cdev, kobj);
+ tmp = kobj_remap(map, dev, mode, all ? MINORMASK : 1, NULL,
+   exact_match, exact_lock, c);
+ if (tmp < 0) {
+  cdev_put(c);
+  return tmp;
+ }
+
+ return 0;
+}
+
+int cdev_del_from_map(struct kobj_map *map, dev_t dev, int all)
+{
```

```
+ int tmp;
+ struct kobject *k;
+ struct cdev *c;
+
+ k = kobj_lookup(cdev_map, dev, NULL, &tmp);
+ if (k == NULL)
+  return -ENODEV;
+
+ c = container_of(k, struct cdev, kobj);
+ kobj_unmap(map, dev, all ? MINORMASK : 1);
+
+ cdev_put(c);
+ cdev_put(c);
+ return 0;
+}
+
+void cdev_iterate_map(struct kobj_map *map,
+  int (*fn)(dev_t, int, mode_t, void *), void *x)
+{
+ kobj_map_iterate(map, fn, x);
+}
+#endif
+
 static void cdev_unmap(dev_t dev, unsigned count)
 {
  kobj_unmap(cdev_map, dev, count);
@@ -540,9 +603,19 @@ static struct kobject *base_probe(dev_t dev, int *part, void *data)
  return NULL;
 }

+struct kobj_map *cdev_map_init(void)
+{
+ return kobj_map_init(base_probe, &chrdevs_lock);
+}
+
+void cdev_map_fini(struct kobj_map *map)
+{
+ kobj_map_fini(map);
+}
+
 void __init chrdev_init(void)
 {
- cdev_map = kobj_map_init(base_probe, &chrdevs_lock);
+ cdev_map = cdev_map_init();
  bdi_init(&directly_mappable_cdev_bdi);
 }

diff --git a/include/linux/cdev.h b/include/linux/cdev.h
```

```
index 1e29b13..fe0e560 100644
--- a/include/linux/cdev.h
+++ b/include/linux/cdev.h
@@ -9,6 +9,7 @@
 struct file_operations;
 struct inode;
 struct module;
+struct kobj_map;

 struct cdev {
  struct kobject kobj;
@@ -33,5 +34,11 @@ void cd_forget(struct inode *);

 extern struct backing_dev_info directly_mappable_cdev_bdi;

+int cdev_add_to_map(struct kobj_map *map, dev_t dev, int all, mode_t mode);
+int cdev_del_from_map(struct kobj_map *map, dev_t dev, int all);
+struct kobj_map *cdev_map_init(void);
+void cdev_map_fini(struct kobj_map *map);
+void cdev_iterate_map(struct kobj_map *,
+  int (*fn)(dev_t, int, mode_t, void *), void *);
 #endif
 #endif
```