Subject: [PATCH 1/4] Some changes in the kobject mapper
Posted by Pavel Emelianov on Thu, 07 Feb 2008 12:57:23 GMT
View Forum Message <> Reply to Message

The main thing that I want from the kobj mapper is to add
the mode_t on the struct kobj_map that reflects with
permissions are associated with this particular map. This
mode is to be returned via the kobj_lookup.

I use the FMODE_XXX flags to handle the permissions bits,
as I will compare these ones to the file->f_mode later.
By default all bits are set (for the initial container).

The additional things I need are kobj_remap() to change
that permission and kobj_iterate() to walk the map.

The kobj_map_fini() is the roll-back for the kobj_map_init().

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

```
diff --git a/drivers/base/map.c b/drivers/base/map.c
index e87017f..1aa2b58 100644
--- a/drivers/base/map.c
+++ b/drivers/base/map.c
@@ -15,11 +15,13 @@
 #include <linux/kdev_t.h>
 #include <linux/kobject.h>
 #include <linux/kobj_map.h>
+#include <linux/fs.h>

 struct kobj_map {
  struct probe {
   struct probe *next;
   dev_t dev;
+  mode_t mode;
   unsigned long range;
   struct module *owner;
   kobj_probe_t *get;
@@ -29,9 +31,9 @@ struct kobj_map {
  struct mutex *lock;
 };

-int kobj_map(struct kobj_map *domain, dev_t dev, unsigned long range,
-      struct module *module, kobj_probe_t *probe,
-      int (*lock)(dev_t, void *), void *data)
+static int __kobj_map(struct kobj_map *domain, dev_t dev, mode_t mode,
```

```
+  unsigned long range, struct module *module,
+  kobj_probe_t *probe, int (*lock)(dev_t, void *), void *data)
 {
  unsigned n = MAJOR(dev + range - 1) - MAJOR(dev) + 1;
  unsigned index = MAJOR(dev);
@@ -53,8 +55,10 @@ int kobj_map(struct kobj_map *domain, dev_t dev, unsigned long range,
   p->dev = dev;
   p->range = range;
   p->data = data;
+  /* we allow these ones always by default */
+  p->mode = mode | FMODE_LSEEK | FMODE_PREAD | FMODE_PWRITE;
  }
- mutex_lock(domain->lock);
+
  for (i = 0, p -= n; i < n; i++, p++, index++) {
   struct probe **s = &domain->probes[index % 255];
   while (*s && (*s)->range < range)
@@ -62,10 +66,57 @@ int kobj_map(struct kobj_map *domain, dev_t dev, unsigned long range,
   p->next = *s;
   *s = p;
  }
- mutex_unlock(domain->lock);
  return 0;
 }

+int kobj_map(struct kobj_map *domain, dev_t dev, unsigned long range,
+      struct module *module, kobj_probe_t *probe,
+      int (*lock)(dev_t, void *), void *data)
+{
+ int err;
+
+ mutex_lock(domain->lock);
+ err = __kobj_map(domain, dev, FMODE_READ | FMODE_WRITE, range,
+   module, probe, lock, data);
+ mutex_unlock(domain->lock);
+ return err;
+}
+
+#ifdef CONFIG_CGROUP_DEVS
+int kobj_remap(struct kobj_map *domain, dev_t dev, mode_t mode,
+  unsigned long range, struct module *module,
+  kobj_probe_t *probe, int (*lock)(dev_t, void *), void *data)
+{
+ unsigned n = MAJOR(dev + range - 1) - MAJOR(dev) + 1;
+ unsigned index = MAJOR(dev);
+ unsigned i;
+ int err = -ESRCH;
+
```

```
+ if (n > 255)
+  n = 255;
+
+ mutex_lock(domain->lock);
+ for (i = 0; i < n; i++, index++) {
+  struct probe **s;
+  for (s = &domain->probes[index % 255]; *s; s = &(*s)->next) {
+   struct probe *p = *s;
+   if (p->dev == dev) {
+    p->mode = mode | FMODE_LSEEK |
+     FMODE_PREAD | FMODE_PWRITE;
+    err = 0;
+    break;
+   }
+  }
+ }
+
+ if (err)
+  err = __kobj_map(domain, dev, mode, range, module,
+    probe, lock, data);
+ mutex_unlock(domain->lock);
+ return err;
+}
+#endif
+
 void kobj_unmap(struct kobj_map *domain, dev_t dev, unsigned long range)
 {
  unsigned n = MAJOR(dev + range - 1) - MAJOR(dev) + 1;
@@ -93,7 +144,8 @@ void kobj_unmap(struct kobj_map *domain, dev_t dev, unsigned long
range)
  kfree(found);
 }

-struct kobject *kobj_lookup(struct kobj_map *domain, dev_t dev, int *index)
+struct kobject *kobj_lookup(struct kobj_map *domain, dev_t dev, mode_t *mode,
+  int *index)
 {
  struct kobject *kobj;
  struct probe *p;
@@ -125,14 +177,46 @@ retry:
   kobj = probe(dev, index, data);
   /* Currently ->owner protects _only_ ->probe() itself. */
   module_put(owner);
-  if (kobj)
+  if (kobj) {
+   if (mode)
+    *mode = p->mode;
    return kobj;
```

```
+  }
   goto retry;
  }
  mutex_unlock(domain->lock);
  return NULL;
 }

+#ifdef CONFIG_CGROUP_DEVS
+void kobj_map_iterate(struct kobj_map *domain,
+  int (*fn)(dev_t, int, mode_t, void *), void *arg)
+{
+ int i;
+ struct probe *p;
+ dev_t skip = MKDEV(0, 0);
+
+ mutex_lock(domain->lock);
+ for (i = 0; i < 255; i++) {
+  p = domain->probes[i];
+  while (p != NULL) {
+   if (p->dev == skip)
+    goto next;
+   if (p->dev == MKDEV(0, 1))
+    goto next;
+
+   skip = p->dev;
+   if (fn(p->dev, p->range, p->mode, arg))
+    goto done;
+next:
+   p = p->next;
+  }
+ }
+done:
+ mutex_unlock(domain->lock);
+}
+#endif
+
 struct kobj_map *kobj_map_init(kobj_probe_t *base_probe, struct mutex *lock)
 {
  struct kobj_map *p = kmalloc(sizeof(struct kobj_map), GFP_KERNEL);
@@ -153,3 +237,21 @@ struct kobj_map *kobj_map_init(kobj_probe_t *base_probe, struct
mutex *lock)
  p->lock = lock;
  return p;
 }
+
+void kobj_map_fini(struct kobj_map *map)
+{
+ int i;
```

```
+ struct probe *p, *next;
+
+ for (i = 0; i < 256; i++) {
+  p = map->probes[i];
+  while (p->next != NULL) {
+   next = p->next;
+   kfree(p);
+   p = next;
+  }
+ }
+
+ kfree(p);
+ kfree(map);
+}
diff --git a/include/linux/kobj_map.h b/include/linux/kobj_map.h
index bafe178..ecfe772 100644
--- a/include/linux/kobj_map.h
+++ b/include/linux/kobj_map.h
@@ -7,8 +7,13 @@ struct kobj_map;

 int kobj_map(struct kobj_map *, dev_t, unsigned long, struct module *,
       kobj_probe_t *, int (*)(dev_t, void *), void *);
+int kobj_remap(struct kobj_map *, dev_t, mode_t, unsigned long, struct module *,
+      kobj_probe_t *, int (*)(dev_t, void *), void *);
 void kobj_unmap(struct kobj_map *, dev_t, unsigned long);
-struct kobject *kobj_lookup(struct kobj_map *, dev_t, int *);
+struct kobject *kobj_lookup(struct kobj_map *, dev_t, mode_t *, int *);
+void kobj_map_iterate(struct kobj_map *, int (*fn)(dev_t, int, mode_t, void *),
+  void *);
 struct kobj_map *kobj_map_init(kobj_probe_t *, struct mutex *);
+void kobj_map_fini(struct kobj_map *);

 #endif
```

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