Subject: [PATCH 06/25] sysfs: Simplify readdir.
Posted by ebiederm on Tue, 07 Aug 2007 21:13:14 GMT
View Forum Message <> Reply to Message

At some point someone wrote sysfs_readdir to insert a cursor
into the list of sysfs_dirents to ensure that sysfs_readdir would
restart properly. That works but it is complex code and tends
to be expensive.

The same effect can be achieved by keeping the sysfs_dirents in
inode order and using the inode number as the f_pos. Then
when we restart we just have to find the first dirent whose inode
number is equal or greater then the last sysfs_dirent we attempted
to return.

Removing the sysfs directory cursor also allows the remove of
all of the mysterious checks for sysfs_type(sd) != 0.   Which
were nonbovious checks to see if a cursor was in a directory list.

Signed-off-by: Eric W. Biederman <ebiederm@xmission.com>

```
---
 fs/sysfs/dir.c |  175 +++++++++++++-----------------------------------------
 1 files changed, 44 insertions(+), 131 deletions(-)

diff --git a/fs/sysfs/dir.c b/fs/sysfs/dir.c
index 2721e36..ef99883 100644
--- a/fs/sysfs/dir.c
+++ b/fs/sysfs/dir.c
@@ -33,10 +33,20 @@ static DEFINE_IDA(sysfs_ino_ida);
 static void sysfs_link_sibling(struct sysfs_dirent *sd)
 {
  struct sysfs_dirent *parent_sd = sd->s_parent;
+ struct sysfs_dirent **pos;

  BUG_ON(sd->s_sibling);
- sd->s_sibling = parent_sd->s_children;
- parent_sd->s_children = sd;
+
+ /* Store directory entries in order by ino.  This allows
+  * readdir to properly restart without having to add a
+  * cursor into the s_children list.
+  */
+ for (pos = &parent_sd->s_children; *pos; pos = &(*pos)->s_sibling) {
+  if (sd->s_ino < (*pos)->s_ino)
+   break;
+ }
+ sd->s_sibling = *pos;
+ *pos = sd;
```

```
 }

 /**
@@ -657,7 +667,7 @@ struct sysfs_dirent *sysfs_find_dirent(struct sysfs_dirent *parent_sd,
  struct sysfs_dirent *sd;

  for (sd = parent_sd->s_children; sd; sd = sd->s_sibling)
-  if (sysfs_type(sd) && !strcmp(sd->s_name, name))
+  if (!strcmp(sd->s_name, name))
    return sd;
  return NULL;
 }
@@ -809,7 +819,7 @@ static void __sysfs_remove_dir(struct sysfs_dirent *dir_sd)
  while (*pos) {
   struct sysfs_dirent *sd = *pos;

-  if (sysfs_type(sd) && sysfs_type(sd) != SYSFS_DIR)
+  if (sysfs_type(sd) != SYSFS_DIR)
    sysfs_remove_one(&acxt, sd);
   else
    pos = &(*pos)->s_sibling;
@@ -974,37 +984,6 @@ again:
  return error;
 }

-static int sysfs_dir_open(struct inode *inode, struct file *file)
-{
- struct dentry * dentry = file->f_path.dentry;
- struct sysfs_dirent * parent_sd = dentry->d_fsdata;
- struct sysfs_dirent * sd;
-
- sd = sysfs_new_dirent("_DIR_", 0, 0);
- if (sd) {
-  mutex_lock(&sysfs_mutex);
-  sd->s_parent = sysfs_get(parent_sd);
-  sysfs_link_sibling(sd);
-  mutex_unlock(&sysfs_mutex);
- }
-
- file->private_data = sd;
- return sd ? 0 : -ENOMEM;
-}
-
-static int sysfs_dir_close(struct inode *inode, struct file *file)
-{
- struct sysfs_dirent * cursor = file->private_data;
-
- mutex_lock(&sysfs_mutex);
```

```
- sysfs_unlink_sibling(cursor);
- mutex_unlock(&sysfs_mutex);
-
- release_sysfs_dirent(cursor);
-
- return 0;
-}
-
 /* Relationship between s_mode and the DT_xxx types */
 static inline unsigned char dt_type(struct sysfs_dirent *sd)
 {
@@ -1015,117 +994,51 @@ static int sysfs_readdir(struct file * filp, void * dirent, filldir_t filldir)
 {
  struct dentry *dentry = filp->f_path.dentry;
  struct sysfs_dirent * parent_sd = dentry->d_fsdata;
- struct sysfs_dirent *cursor = filp->private_data;
- struct sysfs_dirent **pos;
+ struct sysfs_dirent *pos;
  ino_t ino;
- int i = filp->f_pos;

- switch (i) {
-  case 0:
-   ino = parent_sd->s_ino;
-   if (filldir(dirent, ".", 1, i, ino, DT_DIR) < 0)
-    break;
+ if (filp->f_pos == 0) {
+  ino = parent_sd->s_ino;
+  if (filldir(dirent, ".", 1, filp->f_pos, ino, DT_DIR) == 0)
    filp->f_pos++;
-   i++;
-   /* fallthrough */
-  case 1:
-   if (parent_sd->s_parent)
-    ino = parent_sd->s_parent->s_ino;
-   else
-    ino = parent_sd->s_ino;
-   if (filldir(dirent, "..", 2, i, ino, DT_DIR) < 0)
-    break;
+ }
+ if (filp->f_pos == 1) {
+  if (parent_sd->s_parent)
+   ino = parent_sd->s_parent->s_ino;
+  else
+   ino = parent_sd->s_ino;
+  if (filldir(dirent, "..", 2, filp->f_pos, ino, DT_DIR) == 0)
    filp->f_pos++;
-   i++;
```

```
-   /* fallthrough */
-  default:
-   mutex_lock(&sysfs_mutex);
-
-   pos = &parent_sd->s_children;
-   while (*pos != cursor)
-    pos = &(*pos)->s_sibling;
-
-   /* unlink cursor */
-   *pos = cursor->s_sibling;
-
-   if (filp->f_pos == 2)
-    pos = &parent_sd->s_children;
-
-   for ( ; *pos; pos = &(*pos)->s_sibling) {
-    struct sysfs_dirent *next = *pos;
-    const char * name;
-    int len;
-
-    if (!sysfs_type(next))
-     continue;
-
-    name = next->s_name;
-    len = strlen(name);
-    ino = next->s_ino;
-
-    if (filldir(dirent, name, len, filp->f_pos, ino,
-       dt_type(next)) < 0)
-     break;
-
-    filp->f_pos++;
-   }
+ }
+ if ((filp->f_pos > 1) && (filp->f_pos < UINT_MAX)) {
+  mutex_lock(&sysfs_mutex);

-   /* put cursor back in */
-   cursor->s_sibling = *pos;
-   *pos = cursor;
+  /* Skip the dentries we have already reported */
+  pos = parent_sd->s_children;
+  while (pos && (filp->f_pos > pos->s_ino))
+   pos = pos->s_sibling;

-   mutex_unlock(&sysfs_mutex);
- }
- return 0;
-}
```

```
+  for ( ; pos; pos = pos->s_sibling) {
+   const char * name;
+   int len;

-static loff_t sysfs_dir_lseek(struct file * file, loff_t offset, int origin)
-{
- struct dentry * dentry = file->f_path.dentry;
+   name = pos->s_name;
+   len = strlen(name);
+   filp->f_pos = ino = pos->s_ino;

- switch (origin) {
-  case 1:
-   offset += file->f_pos;
-  case 0:
-   if (offset >= 0)
+   if (filldir(dirent, name, len, filp->f_pos, ino,
+       dt_type(pos)) < 0)
     break;
-  default:
-   return -EINVAL;
- }
- if (offset != file->f_pos) {
-  mutex_lock(&sysfs_mutex);
-
-  file->f_pos = offset;
-  if (file->f_pos >= 2) {
-   struct sysfs_dirent *sd = dentry->d_fsdata;
-   struct sysfs_dirent *cursor = file->private_data;
-   struct sysfs_dirent **pos;
-   loff_t n = file->f_pos - 2;
-
-   sysfs_unlink_sibling(cursor);
-
-   pos = &sd->s_children;
-   while (n && *pos) {
-    struct sysfs_dirent *next = *pos;
-    if (sysfs_type(next))
-     n--;
-    pos = &(*pos)->s_sibling;
-   }
-
-   cursor->s_sibling = *pos;
-   *pos = cursor;
   }
-
+  if (!pos)
+   filp->f_pos = UINT_MAX;
```

```
   mutex_unlock(&sysfs_mutex);
  }
-
- return offset;
+ return 0;
 }

+
 const struct file_operations sysfs_dir_operations = {
- .open  = sysfs_dir_open,
- .release = sysfs_dir_close,
- .llseek  = sysfs_dir_lseek,
  .read  = generic_read_dir,
  .readdir = sysfs_readdir,
 };
--
1.5.1.1.181.g2de0
```

_____

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