Subject: [patch 02/22] filesystem helpers for custom struct files
Posted by Cedric Le Goater on Thu, 07 Jun 2007 15:25:29 GMT
View Forum Message <> Reply to Message

From: Dave Hansen <hansendc@us.ibm.com>

Christoph H. says this stands on its own and can go in before the
rest of the r/o bind mount set.

Some filesystems forego the vfs and may_open() and create their
own 'struct file's.

This patch creates a couple of helper functions which can be
used by these filesystems, and will provide a unified place
which the r/o bind mount code may patch.

Also, rename two existing, static-scope init_file() to less
generic names.

Signed-off-by: Dave Hansen <hansendc@us.ibm.com>
---
 fs/configfs/dir.c | 5 +++--
 fs/file_table.c | 36 ++++++++++++++++++++++++++++++++++++
 fs/hugetlbfs/inode.c | 22 +++++++++-------------
 include/linux/file.h | 9 +++++++++
 mm/shmem.c | 7 ++-----
 mm/tiny-shmem.c | 19 +++++++------------
 net/socket.c | 18 +++++++++---------
 7 files changed, 76 insertions(+), 40 deletions(-)

Index: 2.6.22-rc4-mm2-robindmount/fs/configfs/dir.c
===
--- 2.6.22-rc4-mm2-robindmount.orig/fs/configfs/dir.c
+++ 2.6.22-rc4-mm2-robindmount/fs/configfs/dir.c
@@ -142,7 +142,7 @@ static int init_dir(struct inode * inode
  return 0;
 }

-static int init_file(struct inode * inode)
+static int configfs_init_file(struct inode * inode)
 {
  inode->i_size = PAGE_SIZE;
  inode->i_fop = &configfs_file_operations;
@@ -283,7 +283,8 @@ static int configfs_attach_attr(struct c

  dentry->d_fsdata = configfs_get(sd);
  sd->s_dentry = dentry;
- error = configfs_create(dentry, (attr->ca_mode & S_IALLUGO) | S_IFREG, init_file);

```
+ error = configfs_create(dentry, (attr->ca_mode & S_IALLUGO) | S_IFREG,
+     configfs_init_file);
  if (error) {
   configfs_put(sd);
   return error;
Index: 2.6.22-rc4-mm2-robindmount/fs/file_table.c
===================================================================
--- 2.6.22-rc4-mm2-robindmount.orig/fs/file_table.c
+++ 2.6.22-rc4-mm2-robindmount/fs/file_table.c
@@ -138,6 +138,42 @@ fail:

 EXPORT_SYMBOL(get_empty_filp);

+struct file *alloc_file(struct vfsmount *mnt, struct dentry *dentry,
+  mode_t mode, const struct file_operations *fop)
+{
+ struct file *file;
+ struct path;
+
+ file = get_empty_filp();
+ if (!file)
+  return NULL;
+
+ init_file(file, mnt, dentry, mode, fop);
+ return file;
+}
+
+EXPORT_SYMBOL(alloc_file);
+
+/*
+ * Note: This is a crappy interface.  It is here to make
+ * merging with the existing users of get_empty_filp()
+ * who have complex failure logic easier.  All users
+ * of this should be moving to alloc_file().
+ */
+int init_file(struct file *file, struct vfsmount *mnt, struct dentry *dentry,
+     mode_t mode, const struct file_operations *fop)
+{
+ int error = 0;
+ file->f_path.dentry = dentry;
+ file->f_path.mnt = mntget(mnt);
+ file->f_mapping = dentry->d_inode->i_mapping;
+ file->f_mode = mode;
+ file->f_op = fop;
+ return error;
+}
+
+EXPORT_SYMBOL(init_file);
```

```
+
 void fastcall fput(struct file *file)
 {
  if (atomic_dec_and_test(&file->f_count))
Index: 2.6.22-rc4-mm2-robindmount/fs/hugetlbfs/inode.c
===================================================================
--- 2.6.22-rc4-mm2-robindmount.orig/fs/hugetlbfs/inode.c
+++ 2.6.22-rc4-mm2-robindmount/fs/hugetlbfs/inode.c
@@ -768,16 +768,11 @@ struct file *hugetlb_zero_setup(size_t s
  if (!dentry)
   goto out_shm_unlock;

- error = -ENFILE;
- file = get_empty_filp();
- if (!file)
-  goto out_dentry;
-
  error = -ENOSPC;
  inode = hugetlbfs_get_inode(root->d_sb, current->fsuid,
     current->fsgid, S_IFREG | S_IRWXUGO, 0);
  if (!inode)
-  goto out_file;
+  goto out_dentry;

  error = -ENOMEM;
  if (hugetlb_reserve_pages(inode, 0, size >> HPAGE_SHIFT))
@@ -786,17 +781,18 @@ struct file *hugetlb_zero_setup(size_t s
  d_instantiate(dentry, inode);
  inode->i_size = size;
  inode->i_nlink = 0;
- file->f_path.mnt = mntget(hugetlbfs_vfsmount);
- file->f_path.dentry = dentry;
- file->f_mapping = inode->i_mapping;
- file->f_op = &hugetlbfs_file_operations;
- file->f_mode = FMODE_WRITE | FMODE_READ;
+
+ error = -ENFILE;
+ file = alloc_file(hugetlbfs_vfsmount, dentry,
+   FMODE_WRITE | FMODE_READ,
+   &hugetlbfs_file_operations);
+ if (!file)
+  goto out_inode;
+
  return file;

 out_inode:
  iput(inode);
-out_file:
```

```
- put_filp(file);
 out_dentry:
  dput(dentry);
 out_shm_unlock:
Index: 2.6.22-rc4-mm2-robindmount/include/linux/file.h
===================================================================
--- 2.6.22-rc4-mm2-robindmount.orig/include/linux/file.h
+++ 2.6.22-rc4-mm2-robindmount/include/linux/file.h
@@ -62,6 +62,15 @@ extern struct kmem_cache *filp_cachep;
 extern void FASTCALL(__fput(struct file *));
 extern void FASTCALL(fput(struct file *));

+struct file_operations;
+struct vfsmount;
+struct dentry;
+extern int init_file(struct file *, struct vfsmount *mnt,
+  struct dentry *dentry, mode_t mode,
+  const struct file_operations *fop);
+extern struct file *alloc_file(struct vfsmount *, struct dentry *dentry,
+  mode_t mode, const struct file_operations *fop);
+
 static inline void fput_light(struct file *file, int fput_needed)
 {
  if (unlikely(fput_needed))
Index: 2.6.22-rc4-mm2-robindmount/mm/shmem.c
===================================================================
--- 2.6.22-rc4-mm2-robindmount.orig/mm/shmem.c
+++ 2.6.22-rc4-mm2-robindmount/mm/shmem.c
@@ -2503,11 +2503,8 @@ struct file *shmem_file_setup(char *name
  d_instantiate(dentry, inode);
  inode->i_size = size;
  inode->i_nlink = 0; /* It is unlinked */
- file->f_path.mnt = mntget(shm_mnt);
- file->f_path.dentry = dentry;
- file->f_mapping = inode->i_mapping;
- file->f_op = &shmem_file_operations;
- file->f_mode = FMODE_WRITE | FMODE_READ;
+ init_file(file, shm_mnt, dentry, FMODE_WRITE | FMODE_READ,
+   &shmem_file_operations);
  return file;

 close_file:
Index: 2.6.22-rc4-mm2-robindmount/mm/tiny-shmem.c
===================================================================
--- 2.6.22-rc4-mm2-robindmount.orig/mm/tiny-shmem.c
+++ 2.6.22-rc4-mm2-robindmount/mm/tiny-shmem.c
@@ -66,24 +66,19 @@ struct file *shmem_file_setup(char *name
  if (!dentry)
```

```
   goto put_memory;

- error = -ENFILE;
- file = get_empty_filp();
- if (!file)
-  goto put_dentry;
-
  error = -ENOSPC;
  inode = ramfs_get_inode(root->d_sb, S_IFREG | S_IRWXUGO, 0);
  if (!inode)
-  goto close_file;
+  goto put_dentry;

  d_instantiate(dentry, inode);
- inode->i_nlink = 0; /* It is unlinked */
+ error = -ENFILE;
+ file = alloc_file(shm_mnt, dentry, FMODE_WRITE | FMODE_READ,
+    &ramfs_file_operations);
+ if (!file)
+  goto put_dentry;

- file->f_path.mnt = mntget(shm_mnt);
- file->f_path.dentry = dentry;
- file->f_mapping = inode->i_mapping;
- file->f_op = &ramfs_file_operations;
- file->f_mode = FMODE_WRITE | FMODE_READ;
+ inode->i_nlink = 0; /* It is unlinked */

  /* notify everyone as to the change of file size */
  error = do_truncate(dentry, size, 0, file);
Index: 2.6.22-rc4-mm2-robindmount/net/socket.c
===================================================================
--- 2.6.22-rc4-mm2-robindmount.orig/net/socket.c
+++ 2.6.22-rc4-mm2-robindmount/net/socket.c
@@ -364,26 +364,28 @@ static int sock_alloc_fd(struct file **f

 static int sock_attach_fd(struct socket *sock, struct file *file)
 {
+ struct dentry *dentry;
  struct qstr name = { .name = "" };
+ struct path;

- file->f_path.dentry = d_alloc(sock_mnt->mnt_sb->s_root, &name);
- if (unlikely(!file->f_path.dentry))
+ dentry = d_alloc(sock_mnt->mnt_sb->s_root, &name);
+ if (unlikely(!dentry))
   return -ENOMEM;
```

```
- file->f_path.dentry->d_op = &sockfs_dentry_operations;
+ dentry->d_op = &sockfs_dentry_operations;
  /*
   * We dont want to push this dentry into global dentry hash table.
   * We pretend dentry is already hashed, by unsetting DCACHE_UNHASHED
   * This permits a working /proc/$pid/fd/XXX on sockets
   */
- file->f_path.dentry->d_flags &= ~DCACHE_UNHASHED;
- d_instantiate(file->f_path.dentry, SOCK_INODE(sock));
- file->f_path.mnt = mntget(sock_mnt);
- file->f_mapping = file->f_path.dentry->d_inode->i_mapping;
+ dentry->d_flags &= ~DCACHE_UNHASHED;
+ d_instantiate(dentry, SOCK_INODE(sock));
+ init_file(file, sock_mnt, dentry, FMODE_READ | FMODE_WRITE,
+  &socket_file_ops);
+ SOCK_INODE(sock)->i_fop = &socket_file_ops;

  sock->file = file;
  file->f_op = SOCK_INODE(sock)->i_fop = &socket_file_ops;
- file->f_mode = FMODE_READ | FMODE_WRITE;
  file->f_flags = O_RDWR;
  file->f_pos = 0;
  file->private_data = sock;

--
```