Subject: [PATCH 01/25] filesystem helpers for custom 'struct file's
Posted by Dave Hansen on Mon, 11 Dec 2006 22:30:02 GMT
View Forum Message <> Reply to Message

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.

Signed-off-by: Dave Hansen <haveblue@us.ibm.com>

```
 lxc-dave/fs/file_table.c      |   30 ++++++++++++++++++++++++++++++
 lxc-dave/fs/hugetlbfs/inode.c |   22 +++++++++----------
 lxc-dave/include/linux/file.h |    8 ++++++++
 lxc-dave/mm/shmem.c           |    7 ++-----
 lxc-dave/mm/tiny-shmem.c      |   24 +++++++++++++----------
 lxc-dave/net/socket.c         |   18 +++++++++---------
 6 files changed, 67 insertions(+), 42 deletions(-)

diff -puN fs/file_table.c~01-24-filesystem-helpers-for-custom-struct-file-s fs/file_table.c
--- lxc/fs/file_table.c~01-24-filesystem-helpers-for-custom-struct-file-s 2006-12-11
14:21:56.000000000 -0800
+++ lxc-dave/fs/file_table.c 2006-12-11 14:21:56.000000000 -0800
@@ -139,6 +139,36 @@ 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;
+
+ file = get_empty_filp();
+ if (!file)
+  return NULL;
+
+ init_file(file, mnt, dentry, mode, fop);
+ return file;
+}
+
+EXPORT_SYMBOL(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_vfsmnt = mntget(mnt);
+ file->f_dentry = dentry;
+ 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))
diff -puN fs/hugetlbfs/inode.c~01-24-filesystem-helpers-for-custom-struct-file-s fs/hugetlbfs/inode.c
--- lxc/fs/hugetlbfs/inode.c~01-24-filesystem-helpers-for-custom-struct-file-s 2006-12-11
14:21:56.000000000 -0800
+++ lxc-dave/fs/hugetlbfs/inode.c 2006-12-11 14:21:56.000000000 -0800
@@ -756,16 +756,11 @@ 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))
@@ -774,17 +769,18 @@ 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:
diff -puN include/linux/file.h~01-24-filesystem-helpers-for-custom-struct-file-s include/linux/file.h
--- lxc/include/linux/file.h~01-24-filesystem-helpers-for-custom-struct-file-s 2006-12-11
14:21:56.000000000 -0800
+++ lxc-dave/include/linux/file.h 2006-12-11 14:21:56.000000000 -0800
@@ -62,6 +62,14 @@ 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 *, 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))
diff -puN mm/shmem.c~01-24-filesystem-helpers-for-custom-struct-file-s mm/shmem.c
--- lxc/mm/shmem.c~01-24-filesystem-helpers-for-custom-struct-file-s 2006-12-11
14:21:56.000000000 -0800
+++ lxc-dave/mm/shmem.c 2006-12-11 14:21:56.000000000 -0800
@@ -2493,11 +2493,8 @@ 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:
diff -puN mm/tiny-shmem.c~01-24-filesystem-helpers-for-custom-struct-file-s mm/tiny-shmem.c
--- lxc/mm/tiny-shmem.c~01-24-filesystem-helpers-for-custom-struct-file-s 2006-12-11
14:21:56.000000000 -0800
+++ lxc-dave/mm/tiny-shmem.c 2006-12-11 14:21:56.000000000 -0800
@@ -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_inode;

- 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);
@@ -91,9 +86,8 @@ struct file *shmem_file_setup(char *name
   goto close_file;

  return file;
-
-close_file:
- put_filp(file);
+put_inode:
```

```
+ iput(inode);
 put_dentry:
  dput(dentry);
 put_memory:
diff -puN net/socket.c~01-24-filesystem-helpers-for-custom-struct-file-s net/socket.c
--- lxc/net/socket.c~01-24-filesystem-helpers-for-custom-struct-file-s 2006-12-11
14:21:56.000000000 -0800
+++ lxc-dave/net/socket.c 2006-12-11 14:21:56.000000000 -0800
@@ -355,6 +355,7 @@ static int sock_alloc_fd(struct file **f

 static int sock_attach_fd(struct socket *sock, struct file *file)
 {
+ struct dentry *dentry;
  struct qstr this;
  char name[32];

@@ -362,24 +363,23 @@ static int sock_attach_fd(struct socket
  this.name = name;
  this.hash = 0;

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

–

_____

Containers mailing list
Containers@lists.osdl.org
https://lists.osdl.org/mailman/listinfo/containers