Subject: [PATCH 25/25] add kerneldoc for i_nlink helpers
Posted by Dave Hansen on Mon, 11 Dec 2006 22:30:20 GMT
View Forum Message <> Reply to Message

I left this until the end of the series because some of the
descriptions here really don't make any sense until this
series is complete.

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

 lxc-dave/include/linux/fs.h | 44 ++
 1 file changed, 44 insertions(+)

diff -puN include/linux/fs.h~24-24-add-kerneldoc-for-i-nlink-helpers include/linux/fs.h
--- lxc/include/linux/fs.h~24-24-add-kerneldoc-for-i-nlink-helpers 2006-12-11 14:22:13.000000000 -0800
+++ lxc-dave/include/linux/fs.h 2006-12-11 14:22:13.000000000 -0800
@@ -1242,6 +1242,14 @@ static inline void mark_inode_dirty_sync
  __mark_inode_dirty(inode, I_DIRTY_SYNC);
 }

+/**
+ * inc_nlink - directly increment an inode's link count
+ * @inode: inode
+ *
+ * This is a low-level filesystem helper to replace any
+ * direct filesystem manipulation of i_nlink. Currently,
+ * it is only here for parity with dec_nlink().
+ */
 static inline void inc_nlink(struct inode *inode)
 {
  inode->i_nlink++;
@@ -1253,6 +1261,18 @@ static inline void inode_inc_link_count(
  mark_inode_dirty(inode);
 }

+/**
+ * check_nlink - check an inode's status after direct
+ *   i_nlink modification.
+ * @inode: inode
+ *
+ * Some filesystems can not make simple incremental changes
+ * to i_nlink, most notably clustered ones. They must do
+ * direct manipulation of i_nlink. This function must be
+ * called after such modifications are complete to make
+ * sure that the VFS knows that the inode is going to go
+ * away.

```
+ */
 static inline void check_nlink(struct inode *inode)
 {
  if (inode->i_nlink)
@@ -1262,12 +1282,36 @@ static inline void check_nlink(struct in
  atomic_inc(&inode->i_sb->s_mnt_writers);
 }

+/**
+ * drop_nlink - directly drop an inode's link count
+ * @inode: inode
+ *
+ * This is a low-level filesystem helper to replace any
+ * direct filesystem manipulation of i_nlink.  In cases
+ * where we are attempting to track writes to the
+ * filesystem, a decrement to zero means an imminent
+ * write when the file is truncated and actually unlinked
+ * on the filesystem.
+ */
 static inline void drop_nlink(struct inode *inode)
 {
  inode->i_nlink--;
  check_nlink(inode);
 }

+/**
+ * clear_nlink - directly zero an inode's link count
+ * @inode: inode
+ *
+ * This is a low-level filesystem helper to replace any
+ * direct filesystem manipulation of i_nlink.  See
+ * drop_nlink() for why we care about i_nlink hitting zero.
+ *
+ * Note that we could do the i_state flag directly in here,
+ * but we call check_nlink() to keep the number of places
+ * where the flag is set to exactly one.  The compiler
+ * should get rid of the superfluous i_nlink check.
+ */
 static inline void clear_nlink(struct inode *inode)
 {
  inode->i_nlink = 0;
_
```