

[PATCH 02/21] Unionfs: Coding style fixes

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-05/msg10702.html>

- *From:* "Josef 'Jeff' Sipek" <jsipek@xxxxxxxxxxxxxx>
 - *Date:* Wed, 23 May 2007 20:35:52 -0400
-

From: Erez Zadok <ezk@xxxxxxxxxxxxxx>

Includes:

- indentation fixes
- 80 column wrapping
- removing unnecessary braces
- removing trailing whitespace

Signed-off-by: Erez Zadok <ezk@xxxxxxxxxxxxxx>

Signed-off-by: Josef 'Jeff' Sipek <jsipek@xxxxxxxxxxxxxx>

```

----
fs/stack.c | 2 +-
fs/unionfs/branchman.c | 3 +-
fs/unionfs/commonfops.c | 75 ++++++++-----
fs/unionfs/copyup.c | 82 ++++++++-----
fs/unionfs/dentry.c | 45 ++++++-----
fs/unionfs/dirfops.c | 15 +++--
fs/unionfs/dirhelper.c | 26 +++++----
fs/unionfs/fanout.h | 42 ++++++----
fs/unionfs/file.c | 6 +-
fs/unionfs/inode.c | 162 ++++++++-----
fs/unionfs/lookup.c | 50 ++++++-----
fs/unionfs/main.c | 41 ++++++-----
fs/unionfs/rdstate.c | 28 +++++----
fs/unionfs/rename.c | 65 ++++++++-----
fs/unionfs/sioq.c | 5 +-
fs/unionfs/sioq.h | 1 -
fs/unionfs/subr.c | 23 +++-----
fs/unionfs/super.c | 31 +++++----
fs/unionfs/union.h | 45 ++++++-----
fs/unionfs/unlink.c | 3 +-
fs/unionfs/xattr.c | 12 +++--
include/linux/fs_stack.h | 5 +-
22 files changed, 439 insertions(+), 328 deletions(-)

```

```

diff --git a/fs/stack.c b/fs/stack.c
index 67716f6..9aee8fc 100644
--- a/fs/stack.c
+++ b/fs/stack.c

```

[PATCH 02/21] Unionfs: Coding style fixes

```
@@ -18,7 +18,7 @@ EXPORT_SYMBOL_GPL(fsstack_copy_inode_size);
* copying
*/
void fsstack_copy_attr_all(struct inode *dest, const struct inode *src,
- int (*get_nlinks)(struct inode *))
+ int (*get_nlinks)(struct inode *))
{
dest->i_mode = src->i_mode;
dest->i_uid = src->i_uid;
diff --git a/fs/unionfs/branchman.c b/fs/unionfs/branchman.c
index 6912be9..eba2221 100644
--- a/fs/unionfs/branchman.c
+++ b/fs/unionfs/branchman.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003-2007 Stony Brook University
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -57,4 +57,3 @@ out:
unionfs_unlock_dentry(dentry);
return err < 0 ? err : bend;
}
-
diff --git a/fs/unionfs/commonfops.c b/fs/unionfs/commonfops.c
index c9df99d..9cf6b81 100644
--- a/fs/unionfs/commonfops.c
+++ b/fs/unionfs/commonfops.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003-2007 Stony Brook University
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -39,7 +39,7 @@ static int copyup_deleted_file(struct file *file, struct dentry *dentry,
hidden_dentry = unionfs_lower_dentry_idx(dentry, bstart);

sprintf(name, ".unionfs%*.1x",
- i_inosize, i_inosize, hidden_dentry->d_inode->i_ino);
+ i_inosize, i_inosize, hidden_dentry->d_inode->i_ino);

/*
* Loop, looking for an unused temp name to copyup to.
@@ -59,7 +59,7 @@ static int copyup_deleted_file(struct file *file, struct dentry *dentry,
sprintf(suffix, "%*.1x", countersize, countersize, counter);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
printk(KERN_DEBUG "unionfs: trying to rename %s to %s\n",
- dentry->d_name.name, name);
+ dentry->d_name.name, name);

tmp_dentry = lookup_one_len(name, hidden_dentry->d_parent,
UNIONFS_TMPNAM_LEN);
@@ -129,15 +129,15 @@ static void cleanup_file(struct file *file)
int i; /* holds (possibly) updated branch index */
i = find_new_branch_index(file, bindex, sb);
if (i < 0)
- printk(KERN_ERR "unionfs: no superlock for file %p\n",
- file);
+ printk(KERN_ERR "unionfs: no superlock for "
+ "file %p\n", file);
else {
unionfs_read_lock(sb);
branchput(sb, i);
unionfs_read_unlock(sb);
- /* XXX: is it correct to use sb->s_root here? */
+ /* XXX: is it OK to use sb->s_root here? */
unionfs_mntput(sb->s_root, i);
- /* XXX: mntget b/c fput below will call mntput */
+ /* mntget b/c fput below will call mntput */
unionfs_mntget(sb->s_root, bindex);
}
fput(unionfs_lower_file_idx(file, bindex));
@@ -174,9 +174,10 @@ static int open_all_files(struct file *file)
branchget(sb, bindex);
unionfs_read_unlock(sb);

- hidden_file = dentry_open(hidden_dentry,
- unionfs_lower_mnt_idx(dentry, bindex),
- file->f_flags);
+ hidden_file =
+ dentry_open(hidden_dentry,
+ unionfs_lower_mnt_idx(dentry, bindex),
+ file->f_flags);
if (IS_ERR(hidden_file)) {
err = PTR_ERR(hidden_file);
goto out;
@@ -211,7 +212,8 @@ static int open_highest_file(struct file *file, int willwrite)
break;
}
atomic_set(&UNIONFS_F(file)->generation,
- atomic_read(&UNIONFS_I(dentry->d_inode)->generation));
+ atomic_read(&UNIONFS_I(dentry->d_inode)->
+ generation));
goto out;
}
```

[PATCH 02/21] Unionfs: Coding style fixes

```
@@ -221,7 +223,8 @@ static int open_highest_file(struct file *file, int willwrite)
branchget(sb, bstart);
unionfs_read_unlock(sb);
hidden_file = dentry_open(hidden_dentry,
- unionfs_lower_mnt_idx(dentry, bstart), file->f_flags);
+ unionfs_lower_mnt_idx(dentry, bstart),
+ file->f_flags);
if (IS_ERR(hidden_file)) {
err = PTR_ERR(hidden_file);
goto out;
@@ -249,9 +252,10 @@ static int do_delayed_copyup(struct file *file, struct dentry *dentry)
for (bindex = bstart - 1; bindex >= 0; bindex--) {
if (!d_deleted(file->f_dentry))
err = copyup_file(parent_inode, file, bstart,
- bindex, inode_size);
+ bindex, inode_size);
else
- err = copyup_deleted_file(file, dentry, bstart, bindex);
+ err = copyup_deleted_file(file, dentry, bstart,
+ bindex);

if (!err)
break;
@@ -292,7 +296,8 @@ int unionfs_file_revalidate(struct file *file, int willwrite)
sb = dentry->d_sb;

/* first revalidate the dentry inside struct file */
- if (!__unionfs_d_revalidate_chain(dentry, NULL) && !d_deleted(dentry)) {
+ if (!__unionfs_d_revalidate_chain(dentry, NULL) &&
+ !d_deleted(dentry)) {
err = -ESTALE;
goto out_nofree;
}
@@ -351,7 +356,7 @@ int unionfs_file_revalidate(struct file *file, int willwrite)
!IS_WRITE_FLAG(unionfs_lower_file(file)->f_flags) &&
is_robranch(dentry)) {
printk(KERN_DEBUG "Doing delayed copyup of a read-write "
- "file on a read-only branch.\n");
+ "file on a read-only branch.\n");
err = do_delayed_copyup(file, dentry);
}

@@ -376,15 +381,17 @@ static int __open_dir(struct inode *inode, struct file *file)
bend = fbend(file) = dbend(file->f_dentry);

for (bindex = bstart; bindex <= bend; bindex++) {
- hidden_dentry = unionfs_lower_dentry_idx(file->f_dentry, bindex);
+ hidden_dentry =
+ unionfs_lower_dentry_idx(file->f_dentry, bindex);
if (!hidden_dentry)
continue;
```

[PATCH 02/21] Unionfs: Coding style fixes

```
dget(hidden_dentry);
unionfs_mntget(file->f_dentry, bindex);
hidden_file = dentry_open(hidden_dentry,
- unionfs_lower_mnt_idx(file->f_dentry, bindex),
- file->f_flags);
+ unionfs_lower_mnt_idx(file->f_dentry,
+ bindex),
+ file->f_flags);
if (IS_ERR(hidden_file))
return PTR_ERR(hidden_file);

@@ -415,8 +422,8 @@ static int __open_file(struct inode *inode, struct file *file)
bstart = fbstart(file) = dbstart(file->f_dentry);
bend = fbend(file) = dbend(file->f_dentry);

- /* check for the permission for hidden file. If the error is COPYUP_ERR,
- * copyup the file.
+ /* check for the permission for hidden file. If the error is
+ * COPYUP_ERR, copyup the file.
*/
if (hidden_dentry->d_inode && is_robranch(file->f_dentry)) {
/* if the open will change the file, copy it up otherwise
@@ -428,8 +435,9 @@ static int __open_file(struct inode *inode, struct file *file)

/* copyup the file */
for (bindex = bstart - 1; bindex >= 0; bindex--) {
- err = copyup_file(file->f_dentry->d_parent->d_inode,
- file, bstart, bindex, size);
+ err = copyup_file(
+ file->f_dentry->d_parent->d_inode,
+ file, bstart, bindex, size);
if (!err)
break;
}
@@ -444,9 +452,10 @@ static int __open_file(struct inode *inode, struct file *file)
* otherwise fput() will do an mntput() for us upon file close.
*/
unionfs_mntget(file->f_dentry, bstart);
- hidden_file = dentry_open(hidden_dentry,
- unionfs_lower_mnt_idx(file->f_dentry, bstart),
- hidden_flags);
+ hidden_file =
+ dentry_open(hidden_dentry,
+ unionfs_lower_mnt_idx(file->f_dentry, bstart),
+ hidden_flags);
if (IS_ERR(hidden_file))
return PTR_ERR(hidden_file);

@@ -467,7 +476,8 @@ int unionfs_open(struct inode *inode, struct file *file)
int size;
```

[PATCH 02/21] Unionfs: Coding style fixes

```
unionfs_read_lock(inode->i_sb);
- file->private_data = kzalloc(sizeof(struct unionfs_file_info), GFP_KERNEL);
+ file->private_data =
+ kzalloc(sizeof(struct unionfs_file_info), GFP_KERNEL);
if (!UNIONFS_F(file)) {
err = -ENOMEM;
goto out_nofree;
@@ -624,7 +634,8 @@ long unionfs_ioctl(struct file *file, unsigned int cmd, unsigned long arg)
switch (cmd) {
case UNIONFS_IOCTL_INCGEN:
/* Increment the superblock generation count */
- printk("unionfs: incgen ioctl deprecated; use \"-o remount,incgen\\\"\\n");
+ printk("unionfs: incgen ioctl deprecated; "
+ "use \"-o remount,incgen\\\"\\n");
err = -ENOSYS;
break;

@@ -665,15 +676,17 @@ int unionfs_flush(struct file *file, fl_owner_t id)
for (bindex = bstart; bindex <= bend; bindex++) {
hidden_file = unionfs_lower_file_idx(file, bindex);

- if (hidden_file && hidden_file->f_op && hidden_file->f_op->flush) {
+ if (hidden_file && hidden_file->f_op &&
+ hidden_file->f_op->flush) {
err = hidden_file->f_op->flush(hidden_file, id);
if (err)
goto out_lock;

- /* if there are no more references to the dentry, dput it */
+ /* if there are no more refs to the dentry, dput it */
if (d_deleted(dentry)) {
dput(unionfs_lower_dentry_idx(dentry, bindex));
- unionfs_set_lower_dentry_idx(dentry, bindex, NULL);
+ unionfs_set_lower_dentry_idx(dentry, bindex,
+ NULL);
}
}

diff --git a/fs/unionfs/copyup.c b/fs/unionfs/copyup.c
index 331c6ee..502a40f 100644
--- a/fs/unionfs/copyup.c
+++ b/fs/unionfs/copyup.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003-2007 Stony Brook University
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York*
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
```

[PATCH 02/21] Unionfs: Coding style fixes

```
* it under the terms of the GNU General Public License version 2 as
@@ -85,7 +85,7 @@ static int copyup_xattrs(struct dentry *old_hidden_dentry,
goto out;
name_list += strlen(name_list) + 1;
}
- out:
+out:
name_list = name_list_orig;

if (name_list)
@@ -118,8 +118,8 @@ static int copyup_permissions(struct super_block *sb,
newattrs.ia_mode = i->i_mode;

newattrs.ia_valid = ATTR_CTIME | ATTR_ATIME | ATTR_MTIME |
- ATTR_ATIME_SET | ATTR_MTIME_SET | ATTR_FORCE |
- ATTR_GID | ATTR_UID | ATTR_MODE;
+ ATTR_ATIME_SET | ATTR_MTIME_SET | ATTR_FORCE |
+ ATTR_GID | ATTR_UID | ATTR_MODE;

err = notify_change(new_hidden_dentry, &newattrs);

@@ -184,7 +184,7 @@ static int __copyup_ndentry(struct dentry *old_hidden_dentry,
err = args.err;
} else {
printk(KERN_ERR "Unknown inode type %d\n",
- old_mode);
+ old_mode);
BUG();
}

@@ -211,8 +211,8 @@ static int __copyup_reg_data(struct dentry *dentry,
branchget(sb, old_bindex);
unionfs_read_unlock(sb);
input_file = dentry_open(old_hidden_dentry,
- unionfs_lower_mnt_idx(dentry, old_bindex),
- O_RDONLY | O_LARGEFILE);
+ unionfs_lower_mnt_idx(dentry, old_bindex),
+ O_RDONLY | O_LARGEFILE);
if (IS_ERR(input_file)) {
dput(old_hidden_dentry);
err = PTR_ERR(input_file);
@@ -230,8 +230,8 @@ static int __copyup_reg_data(struct dentry *dentry,
branchget(sb, new_bindex);
unionfs_read_unlock(sb);
output_file = dentry_open(new_hidden_dentry,
- unionfs_lower_mnt_idx(dentry, new_bindex),
- O_WRONLY | O_LARGEFILE);
+ unionfs_lower_mnt_idx(dentry, new_bindex),
+ O_WRONLY | O_LARGEFILE);
if (IS_ERR(output_file)) {
err = PTR_ERR(output_file);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
goto out_close_in2;
@@ -265,19 +265,19 @@ static int __copyup_reg_data(struct dentry *dentry,
len -= PAGE_SIZE;

read_bytes =
- input_file->f_op->read(input_file,
- (char __user *)buf, size,
- &input_file->f_pos);
+ input_file->f_op->read(input_file,
+ (char __user *)buf, size,
+ &input_file->f_pos);
if (read_bytes <= 0) {
err = read_bytes;
break;
}

write_bytes =
- output_file->f_op->write(output_file,
- (char __user *)buf,
- read_bytes,
- &output_file->f_pos);
+ output_file->f_op->write(output_file,
+ (char __user *)buf,
+ read_bytes,
+ &output_file->f_pos);
if ((write_bytes < 0) || (write_bytes < read_bytes)) {
err = write_bytes;
break;
}
@@ -362,7 +362,8 @@ static int copyup_named_dentry(struct inode *dir, struct dentry *dentry,
goto out;

/* Create the directory structure above this dentry. */
- new_hidden_dentry = create_parents_named(dir, dentry, name, new_bindex);
+ new_hidden_dentry =
+ create_parents_named(dir, dentry, name, new_bindex);
if (IS_ERR(new_hidden_dentry)) {
err = PTR_ERR(new_hidden_dentry);
goto out;
}
@@ -387,9 +388,9 @@ static int copyup_named_dentry(struct inode *dir, struct dentry *dentry,
oldfs = get_fs();
set_fs(KERNEL_DS);
err = old_hidden_dentry->d_inode->i_op->readlink(
- old_hidden_dentry,
- (char __user *)symbuf,
- PATH_MAX);
+ old_hidden_dentry,
+ (char __user *)symbuf,
+ PATH_MAX);
set_fs(oldfs);
if (err) {
__clear(dentry, old_hidden_dentry,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
@@ -417,12 +418,14 @@ static int copyup_named_dentry(struct inode *dir, struct dentry *dentry,
/* We actually copyup the file here. */
if (S_ISREG(old_hidden_dentry->d_inode->i_mode))
err = __copyup_reg_data(dentry, new_hidden_dentry, new_bindex,
- old_hidden_dentry, old_bindex, copyup_file, len);
+ old_hidden_dentry, old_bindex,
+ copyup_file, len);
if (err)
goto out_unlink;

/* Set permissions. */
- if ((err = copyup_permissions(sb, old_hidden_dentry, new_hidden_dentry)))
+ if ((err = copyup_permissions(sb, old_hidden_dentry,
+ new_hidden_dentry)))
goto out_unlink;

#ifdef CONFIG_UNION_FS_XATTR
@@ -520,8 +523,8 @@ int copyup_file(struct inode *dir, struct file *file, int bstart,
return err;
}

-/* This function replicates the directory structure upto given dentry
- * in the bindex branch. Can create directory structure recursively to the right
+/* This function replicates the directory structure upto given dentry in the
+ * bindex branch. Can create directory structure recursively to the right
* also.
*/
struct dentry *create_parents(struct inode *dir, struct dentry *dentry,
@@ -584,7 +587,7 @@ static void __set_inode(struct dentry * upper, struct dentry * lower,
int bindex)
{
unionfs_set_lower_inode_idx(upper->d_inode, bindex,
- igrab(lower->d_inode));
+ igrab(lower->d_inode));
if (likely(ibstart(upper->d_inode) > bindex))
ibstart(upper->d_inode) = bindex;
if (likely(ibend(upper->d_inode) < bindex))
@@ -664,7 +667,8 @@ static struct dentry *create_parents_named(struct inode *dir,
unionfs_lock_dentry(parent_dentry);

/* find out the hidden_parent_dentry in the given branch */
- hidden_parent_dentry = unionfs_lower_dentry_idx(parent_dentry, bindex);
+ hidden_parent_dentry =
+ unionfs_lower_dentry_idx(parent_dentry, bindex);

/* grow path table */
if (count == nr_dentry) {
@@ -691,7 +695,8 @@ static struct dentry *create_parents_named(struct inode *dir,
*/
while (1) {
/* get hidden parent dir in the current branch */
```

[PATCH 02/21] Unionfs: Coding style fixes

```
- hidden_parent_dentry = unionfs_lower_dentry_idx(parent_dentry, bindex);
+ hidden_parent_dentry =
+ unionfs_lower_dentry_idx(parent_dentry, bindex);
unionfs_unlock_dentry(parent_dentry);

/* init the values to lookup */
@@ -701,24 +706,29 @@ static struct dentry *create_parents_named(struct inode *dir,
if (child_dentry != dentry) {
/* lookup child in the underlying file system */
hidden_dentry =
- lookup_one_len(childname, hidden_parent_dentry,
- childnamelen);
+ lookup_one_len(childname, hidden_parent_dentry,
+ childnamelen);
if (IS_ERR(hidden_dentry))
goto out;
} else {

/* is the name a whiteout of the childname ?
- * lookup the whiteout child in the underlying file system
+ * lookup the whiteout child in the underlying file
+ * system
*/
hidden_dentry =
- lookup_one_len(name, hidden_parent_dentry,
- strlen(name));
+ lookup_one_len(name, hidden_parent_dentry,
+ strlen(name));
if (IS_ERR(hidden_dentry))
goto out;

- /* Replace the current dentry (if any) with the new one. */
+ /*
+ * Replace the current dentry (if any) with the new
+ * one.
+ */
dput(unionfs_lower_dentry_idx(dentry, bindex));
- unionfs_set_lower_dentry_idx(dentry, bindex, hidden_dentry);
+ unionfs_set_lower_dentry_idx(dentry, bindex,
+ hidden_dentry);

__cleanup_dentry(dentry, bindex, old_bstart, old_bend);
break;
@@ -744,7 +754,8 @@ static struct dentry *create_parents_named(struct inode *dir,

if (!err)
err = copyup_permissions(dir->i_sb,
- child_dentry, hidden_dentry);
+ child_dentry,
+ hidden_dentry);
unlock_dir(hidden_parent_dentry);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
if (err) {
dput(hidden_dentry);
@@ -764,4 +775,3 @@ out:
kfree(path);
return hidden_dentry;
}
-
diff --git a/fs/unionfs/dentry.c b/fs/unionfs/dentry.c
index 9eb143d..067732c 100644
--- a/fs/unionfs/dentry.c
+++ b/fs/unionfs/dentry.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003-2007 Stony Brook University
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -26,7 +26,8 @@
* the child may not yet be valid.
* Returns 1 if valid, 0 otherwise.
*/
-static int __unionfs_d_revalidate_one(struct dentry *dentry, struct nameidata *nd)
+static int __unionfs_d_revalidate_one(struct dentry *dentry,
+ struct nameidata *nd)
{
int valid = 1; /* default is valid (1); invalid is 0. */
struct dentry *hidden_dentry;
@@ -79,7 +80,8 @@ static int __unionfs_d_revalidate_one(struct dentry *dentry, struct nameidata *n
struct dentry *hidden_dentry;
for (bindex = bstart; bindex <= bend; bindex++) {
hidden_dentry =
- unionfs_lower_dentry_idx(dentry, bindex);
+ unionfs_lower_dentry_idx(dentry,
+ bindex);
dput(hidden_dentry);
}
}
@@ -104,9 +106,11 @@ static int __unionfs_d_revalidate_one(struct dentry *dentry, struct nameidata *n
bend = ibend(dentry->d_inode);
if (bstart >= 0) {
struct inode *hidden_inode;
- for (bindex = bstart; bindex <= bend; bindex++) {
+ for (bindex = bstart; bindex <= bend;
+ bindex++) {
hidden_inode =
- unionfs_lower_inode_idx(dentry->d_inode,
+ unionfs_lower_inode_idx(
+ dentry->d_inode,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
bindex);
input(hidden_inode);
}
@@ -119,7 +123,8 @@ static int __unionfs_d_revalidate_one(struct dentry *dentry, struct nameidata *n
mutex_unlock(&dentry->d_inode->i_mutex);
}

- result = unionfs_lookup_backend(dentry, &lowernd, interpose_flag);
+ result = unionfs_lookup_backend(dentry, &lowernd,
+ interpose_flag);
if (result) {
if (IS_ERR(result)) {
valid = 0;
@@ -150,7 +155,8 @@ static int __unionfs_d_revalidate_one(struct dentry *dentry, struct nameidata *n
if (!hidden_dentry || !hidden_dentry->d_op
|| !hidden_dentry->d_op->d_revalidate)
continue;
- if (!hidden_dentry->d_op->d_revalidate(hidden_dentry, &lowernd))
+ if (!hidden_dentry->d_op->d_revalidate(hidden_dentry,
+ &lowernd))
valid = 0;
}

@@ -159,10 +165,10 @@ static int __unionfs_d_revalidate_one(struct dentry *dentry, struct nameidata *n

if (valid) {
fsstack_copy_attr_all(dentry->d_inode,
- unionfs_lower_inode(dentry->d_inode),
- unionfs_get_nlinks);
+ unionfs_lower_inode(dentry->d_inode),
+ unionfs_get_nlinks);
fsstack_copy_inode_size(dentry->d_inode,
- unionfs_lower_inode(dentry->d_inode));
+ unionfs_lower_inode(dentry->d_inode));
}

out:
@@ -195,9 +201,8 @@ int __unionfs_d_revalidate_chain(struct dentry *dentry, struct nameidata *nd)
dtmp = dtmp->d_parent;
dgen = atomic_read(&UNIONFS_D(dtmp)->generation);
}
- if (chain_len == 0) {
+ if (chain_len == 0)
goto out_this; /* shortcut if parents are OK */
- }

/*
* Allocate array of dentries to reval. We could use linked lists,
@@ -236,18 +241,18 @@ int __unionfs_d_revalidate_chain(struct dentry *dentry, struct nameidata *nd)
if (valid && chain_len > 0 &&
sbgen != dgen && chain[i]->d_inode &&
```

[PATCH 02/21] Unionfs: Coding style fixes

```
S_ISDIR(chain[i]->d_inode->i_mode)) {
- for (bindex = saved_bstart; bindex <= saved_bend; bindex++)
+ for (bindex = saved_bstart; bindex <= saved_bend;
+ bindex++)
unionfs_mntput(chain[i], bindex);
}
unionfs_unlock_dentry(chain[i]);

- if (!valid) {
+ if (!valid)
goto out_free;
- }
}

- out_this:
+out_this:
/* finally, lock this dentry and revalidate it */
verify_locked(dentry);
dgen = atomic_read(&UNIONFS_D(dentry)->generation);
@@ -260,15 +265,14 @@ int __unionfs_d_revalidate_chain(struct dentry *dentry, struct nameidata *nd)
unionfs_mntput(dentry, bindex);
}

- out_free:
+out_free:
/* unlock/dput all dentries in chain and return status */
if (chain_len > 0) {
- for (i=0; i<chain_len; i++) {
+ for (i=0; i<chain_len; i++)
dput(chain[i]);
- }
kfree(chain);
}
- out:
+out:
return valid;
}

@@ -330,4 +334,3 @@ struct dentry_operations unionfs_dops = {
.d_revalidate = unionfs_d_revalidate,
.d_release = unionfs_d_release,
};
-
diff --git a/fs/unionfs/dirfops.c b/fs/unionfs/dirfops.c
index 6ff32a0..cab7b6d 100644
--- a/fs/unionfs/dirfops.c
+++ b/fs/unionfs/dirfops.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
```

[PATCH 02/21] Unionfs: Coding style fixes

```
* Copyright (c) 2003–2007 Stony Brook University
- * Copyright (c) 2003–2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003–2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -70,7 +70,10 @@ static int unionfs_filldir(void *dirent, const char *name, int namelen,
verify_rdstable_offset(buf->rdstate);
}
}
- /* If we did fill it, stuff it in our hash, otherwise return an error */
+ /*
+ * If we did fill it, stuff it in our hash, otherwise return an
+ * error.
+ */
if (err) {
buf->filldir_error = err;
goto out;
@@ -200,7 +203,10 @@ static loff_t unionfs_dir_llseek(struct file *file, loff_t offset, int origin)

rdstate = UNIONFS_F(file)->rdstate;

- /* We let users seek to their current position, but not anywhere else. */
+ /*
+ * we let users seek to their current position, but not anywhere
+ * else.
+ */
if (!offset) {
switch (origin) {
case SEEK_SET:
@@ -231,7 +237,7 @@ static loff_t unionfs_dir_llseek(struct file *file, loff_t offset, int origin)
err = -EINVAL;
} else {
rdstate = find_rdstable(file->f_dentry->d_inode,
- offset);
+ offset);
if (rdstate) {
UNIONFS_F(file)->rdstate = rdstate;
err = rdstate->offset;
@@ -264,4 +270,3 @@ struct file_operations unionfs_dir_fops = {
.release = unionfs_file_release,
.flush = unionfs_flush,
};
-
diff --git a/fs/unionfs/dirhelper.c b/fs/unionfs/dirhelper.c
index bd15eb4..cbea6c1 100644
--- a/fs/unionfs/dirhelper.c
+++ b/fs/unionfs/dirhelper.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
```

[PATCH 02/21] Unionfs: Coding style fixes

```
* Copyright (c) 2003–2007 Stony Brook University
– * Copyright (c) 2003–2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003–2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -23,7 +23,7 @@
* hidden directory inode should be locked
*/
int do_delete_whiteouts(struct dentry *dentry, int bindex,
– struct unionfs_dir_state *namelist)
+ struct unionfs_dir_state *namelist)
{
int err = 0;
struct dentry *hidden_dir_dentry = NULL;
@@ -52,7 +52,8 @@ int do_delete_whiteouts(struct dentry *dentry, int bindex,
for (i = 0; !err && i < namelist->size; i++) {
list_for_each(pos, &namelist->list[i]) {
cursor =
– list_entry(pos, struct filldir_node, file_list);
+ list_entry(pos, struct filldir_node,
+ file_list);
/* Only operate on whiteouts in this branch. */
if (cursor->bindex != bindex)
continue;
@@ -61,8 +62,9 @@ int do_delete_whiteouts(struct dentry *dentry, int bindex,

strcpy(p, cursor->name);
hidden_dentry =
– lookup_one_len(name, hidden_dir_dentry,
– cursor->namelen + UNIONFS_WHLEN);
+ lookup_one_len(name, hidden_dir_dentry,
+ cursor->namelen +
+ UNIONFS_WHLEN);
if (IS_ERR(hidden_dentry)) {
err = PTR_ERR(hidden_dentry);
break;
@@ -148,7 +150,8 @@ static int readdir_util_callback(void *dirent, const char *name, int namelen,

buf->filldir_called = 1;

– if (name[0] == '.' && (namelen == 1 || (name[1] == '.' && namelen == 2)))
+ if (name[0] == '.' && (namelen == 1 ||
+ (name[1] == '.' && namelen == 2)))
goto out;

if (namelen > UNIONFS_WHLEN &&
@@ -163,7 +166,10 @@ static int readdir_util_callback(void *dirent, const char *name, int namelen,
if (found)
goto out;
```

[PATCH 02/21] Unionfs: Coding style fixes

```
- /* If it wasn't found and isn't a whiteout, the directory isn't empty. */
+ /*
+ * if it wasn't found and isn't a whiteout, the directory isn't
+ * empty.
+ */
err = -ENOTEMPTY;
if ((buf->mode == RD_CHECK_EMPTY) && !whiteout)
goto out;
@@ -230,8 +236,9 @@ int check_empty(struct dentry *dentry, struct unionfs_dir_state **namelist)
branchget(sb, bindex);
unionfs_read_unlock(sb);
hidden_file =
- dentry_open(hidden_dentry, unionfs_lower_mnt_idx(dentry, bindex),
- O_RDONLY);
+ dentry_open(hidden_dentry,
+ unionfs_lower_mnt_idx(dentry, bindex),
+ O_RDONLY);
if (IS_ERR(hidden_file)) {
err = PTR_ERR(hidden_file);
dput(hidden_dentry);
@@ -273,4 +280,3 @@ out:

return err;
}
-
diff --git a/fs/unionfs/fanout.h b/fs/unionfs/fanout.h
index 9e4a35f..71052a3 100644
--- a/fs/unionfs/fanout.h
+++ b/fs/unionfs/fanout.h
@@ -8,7 +8,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003-2007 Stony Brook University
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -61,12 +61,14 @@ static inline struct file *unionfs_lower_file(const struct file *f)
return UNIONFS_F(f)->lower_files[fbstart(f)];
}

-static inline struct file *unionfs_lower_file_idx(const struct file *f, int index)
+static inline struct file *unionfs_lower_file_idx(const struct file *f,
+ int index)
{
return UNIONFS_F(f)->lower_files[index];
}

-static inline void unionfs_set_lower_file_idx(struct file *f, int index, struct file *val)
+static inline void unionfs_set_lower_file_idx(struct file *f, int index,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
+ struct file *val)
{
UNIONFS_F(f)->lower_files[index] = val;
/* save branch ID (may be redundant?) */
@@ -85,13 +87,14 @@ static inline struct inode *unionfs_lower_inode(const struct inode *i)
return UNIONFS_I(i)->lower_inodes[ibstart(i)];
}

-static inline struct inode *unionfs_lower_inode_idx(const struct inode *i, int index)
+static inline struct inode *unionfs_lower_inode_idx(const struct inode *i,
+ int index)
{
return UNIONFS_I(i)->lower_inodes[index];
}

static inline void unionfs_set_lower_inode_idx(struct inode *i, int index,
- struct inode *val)
+ struct inode *val)
{
UNIONFS_I(i)->lower_inodes[index] = val;
}
@@ -102,23 +105,28 @@ static inline void unionfs_set_lower_inode(struct inode *i, struct inode *val)
}

/* Superblock to lower superblock. */
-static inline struct super_block *unionfs_lower_super(const struct super_block *sb)
+static inline struct super_block *unionfs_lower_super(
+ const struct super_block *sb)
{
return UNIONFS_SB(sb)->data[sbstart(sb)].sb;
}

-static inline struct super_block *unionfs_lower_super_idx(const struct super_block *sb, int index)
+static inline struct super_block *unionfs_lower_super_idx(
+ const struct super_block *sb,
+ int index)
{
return UNIONFS_SB(sb)->data[index].sb;
}

-static inline void unionfs_set_lower_super_idx(struct super_block *sb, int index,
- struct super_block *val)
+static inline void unionfs_set_lower_super_idx(struct super_block *sb,
+ int index,
+ struct super_block *val)
{
UNIONFS_SB(sb)->data[index].sb = val;
}

-static inline void unionfs_set_lower_super(struct super_block *sb, struct super_block *val)
+static inline void unionfs_set_lower_super(struct super_block *sb,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
+ struct super_block *val)
{
UNIONFS_SB(sb)->data[sbstart(sb)].sb = val;
}
@@ -181,12 +189,14 @@ static inline void set_dbopaque(struct dentry *dent, int val)
}

static inline void unionfs_set_lower_dentry_idx(struct dentry *dent, int index,
- struct dentry *val)
+ struct dentry *val)
{
UNIONFS_D(dent)->lower_paths[index].dentry = val;
}

-static inline struct dentry *unionfs_lower_dentry_idx(const struct dentry *dent, int index)
+static inline struct dentry *unionfs_lower_dentry_idx(
+ const struct dentry *dent,
+ int index)
{
return UNIONFS_D(dent)->lower_paths[index].dentry;
}
@@ -197,19 +207,21 @@ static inline struct dentry *unionfs_lower_dentry(const struct dentry *dent)
}

static inline void unionfs_set_lower_mnt_idx(struct dentry *dent, int index,
- struct vfsmount *mnt)
+ struct vfsmount *mnt)
{
UNIONFS_D(dent)->lower_paths[index].mnt = mnt;
}

-static inline struct vfsmount *unionfs_lower_mnt_idx(const struct dentry *dent, int index)
+static inline struct vfsmount *unionfs_lower_mnt_idx(
+ const struct dentry *dent,
+ int index)
{
return UNIONFS_D(dent)->lower_paths[index].mnt;
}

static inline struct vfsmount *unionfs_lower_mnt(const struct dentry *dent)
{
- return unionfs_lower_mnt_idx(dent,dbstart(dent));
+ return unionfs_lower_mnt_idx(dent, dbstart(dent));
}

/* Macros for locking a dentry. */
diff --git a/fs/unionfs/file.c b/fs/unionfs/file.c
index 84d6bab..430cb43 100644
--- a/fs/unionfs/file.c
+++ b/fs/unionfs/file.c
@@ -9,7 +9,7 @@
```

[PATCH 02/21] Unionfs: Coding style fixes

```
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003–2007 Stony Brook University
– * Copyright (c) 2003–2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003–2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -205,7 +205,8 @@ out:
return err;
}

–static int unionfs_fsync(struct file *file, struct dentry *dentry, int datasync)
+static int unionfs_fsync(struct file *file, struct dentry *dentry,
+ int datasync)
{
int err;
struct file *hidden_file = NULL;
@@ -263,4 +264,3 @@ struct file_operations unionfs_main_fops = {
.fsync = unionfs_fsync,
.fasync = unionfs_fasync,
};
–
diff --git a/fs/unionfs/inode.c b/fs/unionfs/inode.c
index 97dad8c..0b9cb29 100644
--- a/fs/unionfs/inode.c
+++ b/fs/unionfs/inode.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003–2007 Stony Brook University
– * Copyright (c) 2003–2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003–2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -61,7 +61,10 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,
bstart = dbstart(dentry);
hidden_dentry = unionfs_lower_dentry(dentry);

– /* check if whiteout exists in this branch, i.e. lookup .wh.foo first */
+ /*
+ * check if whiteout exists in this branch, i.e. lookup .wh.foo
+ * first.
+ */
name = alloc_whname(dentry->d_name.name, dentry->d_name.len);
if (IS_ERR(name)) {
err = PTR_ERR(name);
@@ -69,7 +72,7 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,
}
```

[PATCH 02/21] Unionfs: Coding style fixes

```
wh_dentry = lookup_one_len(name, hidden_dentry->d_parent,
- dentry->d_name.len + UNIONFS_WHLEN);
+ dentry->d_name.len + UNIONFS_WHLEN);
if (IS_ERR(wh_dentry)) {
err = PTR_ERR(wh_dentry);
wh_dentry = NULL;
@@ -86,8 +89,8 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,

mutex_lock(&wh_dentry->d_inode->i_mutex);
newattrs.ia_valid = ATTR_CTIME | ATTR_MODE | ATTR_ATIME
- | ATTR_MTIME | ATTR_UID | ATTR_GID | ATTR_FORCE
- | ATTR_KILL_SUID | ATTR_KILL_SGID;
+ | ATTR_MTIME | ATTR_UID | ATTR_GID | ATTR_FORCE
+ | ATTR_KILL_SUID | ATTR_KILL_SGID;

newattrs.ia_mode = mode & ~current->fs->umask;
newattrs.ia_uid = current->fsuid;
@@ -104,8 +107,8 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,

if (err)
printk(KERN_WARNING "unionfs: %s:%d: notify_change "
- "failed: %d, ignoring..\n",
- __FILE__, __LINE__, err);
+ "failed: %d, ignoring..\n",
+ __FILE__, __LINE__, err);

new_hidden_dentry = unionfs_lower_dentry(dentry);
dget(new_hidden_dentry);
@@ -121,9 +124,11 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,
}
if (!err) {
fsstack_copy_attr_times(parent,
- new_hidden_dentry->d_parent->d_inode);
+ new_hidden_dentry->d_parent->
+ d_inode);
fsstack_copy_inode_size(parent,
- new_hidden_dentry->d_parent->d_inode);
+ new_hidden_dentry->d_parent->
+ d_inode);
parent->i_nlink = unionfs_get_nlinks(parent);
}

@@ -150,7 +155,8 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,
dput(unionfs_lower_dentry(dentry));

/* Trade one reference to another. */
- unionfs_set_lower_dentry_idx(dentry, bstart, wh_dentry);
+ unionfs_set_lower_dentry_idx(dentry, bstart,
+ wh_dentry);
wh_dentry = NULL;
```

[PATCH 02/21] Unionfs: Coding style fixes

```
err = unionfs_interpose(dentry, parent->i_sb, 0);
@@ -194,10 +200,12 @@ static int unionfs_create(struct inode *parent, struct dentry *dentry,
err = unionfs_interpose(dentry, parent->i_sb, 0);
if (!err) {
fsstack_copy_attr_times(parent,
- hidden_parent_dentry->d_inode);
+ hidden_parent_dentry->
+ d_inode);
fsstack_copy_inode_size(parent,
- hidden_parent_dentry->d_inode);
- /* update number of links on parent directory */
+ hidden_parent_dentry->
+ d_inode);
+ /* update no. of links on parent directory */
parent->i_nlink = unionfs_get_nlinks(parent);
}
unlock_dir(hidden_parent_dentry);
@@ -266,7 +274,8 @@ static int unionfs_link(struct dentry *old_dentry, struct inode *dir,
}

whiteout_dentry = lookup_one_len(name, hidden_new_dentry->d_parent,
- new_dentry->d_name.len + UNIONFS_WHLEN);
+ new_dentry->d_name.len +
+ UNIONFS_WHLEN);
if (IS_ERR(whiteout_dentry)) {
err = PTR_ERR(whiteout_dentry);
goto out;
@@ -281,7 +290,7 @@ static int unionfs_link(struct dentry *old_dentry, struct inode *dir,
err = is_robranch_super(new_dentry->d_sb, dbstart(new_dentry));
if (!err)
err = vfs_unlink(hidden_dir_dentry->d_inode,
- whiteout_dentry);
+ whiteout_dentry);

fsstack_copy_attr_times(dir, hidden_dir_dentry->d_inode);
dir->i_nlink = unionfs_get_nlinks(dir);
@@ -294,7 +303,7 @@ static int unionfs_link(struct dentry *old_dentry, struct inode *dir,

if (dbstart(old_dentry) != dbstart(new_dentry)) {
hidden_new_dentry =
- create_parents(dir, new_dentry, dbstart(old_dentry));
+ create_parents(dir, new_dentry, dbstart(old_dentry));
err = PTR_ERR(hidden_new_dentry);
if (IS_COPYUP_ERR(err))
goto docopyup;
@@ -308,7 +317,7 @@ static int unionfs_link(struct dentry *old_dentry, struct inode *dir,
hidden_dir_dentry = lock_parent(hidden_new_dentry);
if (!(err = is_robranch(old_dentry)))
err = vfs_link(hidden_old_dentry, hidden_dir_dentry->d_inode,
- hidden_new_dentry);
+ hidden_new_dentry);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
unlock_dir(hidden_dir_dentry);

docopyup:
@@ -317,21 +326,22 @@ docopyup:
int bindex;

for (bindex = old_bstart - 1; bindex >= 0; bindex--) {
- err =
- copyup_dentry(old_dentry->d_parent->
- d_inode, old_dentry,
- old_bstart, bindex, NULL,
- old_dentry->d_inode->i_size);
+ err = copyup_dentry(old_dentry->d_parent->d_inode,
+ old_dentry, old_bstart,
+ bindex, NULL,
+ old_dentry->d_inode->i_size);
if (!err) {
hidden_new_dentry =
- create_parents(dir, new_dentry, bindex);
- hidden_old_dentry = unionfs_lower_dentry(old_dentry);
+ create_parents(dir, new_dentry,
+ bindex);
+ hidden_old_dentry =
+ unionfs_lower_dentry(old_dentry);
hidden_dir_dentry =
- lock_parent(hidden_new_dentry);
+ lock_parent(hidden_new_dentry);
/* do vfs_link */
err = vfs_link(hidden_old_dentry,
- hidden_dir_dentry->d_inode,
- hidden_new_dentry);
+ hidden_dir_dentry->d_inode,
+ hidden_new_dentry);
unlock_dir(hidden_dir_dentry);
goto check_link;
}
@@ -347,7 +357,7 @@ check_link:
new_dentry->d_inode = igrab(old_dentry->d_inode);
d_instantiate(new_dentry, new_dentry->d_inode);
fsstack_copy_attr_all(dir, hidden_new_dentry->d_parent->d_inode,
- unionfs_get_nlinks);
+ unionfs_get_nlinks);
fsstack_copy_inode_size(dir, hidden_new_dentry->d_parent->d_inode);

/* propagate number of hard-links */
@@ -395,8 +405,8 @@ static int unionfs_symlink(struct inode *dir, struct dentry *dentry,
}
```

```
whiteout_dentry =
- lookup_one_len(name, hidden_dentry->d_parent,
- dentry->d_name.len + UNIONFS_WHLEN);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
+ lookup_one_len(name, hidden_dentry->d_parent,
+ dentry->d_name.len + UNIONFS_WHLEN);
if (IS_ERR(whiteout_dentry)) {
err = PTR_ERR(whiteout_dentry);
goto out;
@@ -406,12 +416,15 @@ static int unionfs_symlink(struct inode *dir, struct dentry *dentry,
dput(whiteout_dentry);
whiteout_dentry = NULL;
} else {
- /* found a .wh.foo entry, unlink it and then call vfs_symlink() */
+ /*
+ * found a .wh.foo entry, unlink it and then call
+ * vfs_symlink().
+ */
hidden_dir_dentry = lock_parent(whiteout_dentry);

if (!(err = is_robranch_super(dentry->d_sb, bstart)))
err = vfs_unlink(hidden_dir_dentry->d_inode,
- whiteout_dentry);
+ whiteout_dentry);
dput(whiteout_dentry);

fsstack_copy_attr_times(dir, hidden_dir_dentry->d_inode);
@@ -424,7 +437,10 @@ static int unionfs_symlink(struct inode *dir, struct dentry *dentry,
/* exit if the error returned was NOT -EROFS */
if (!IS_COPYUP_ERR(err))
goto out;
- /* should now try to create symlink in the another branch */
+ /*
+ * should now try to create symlink in the another
+ * branch.
+ */
bstart--;
}
}
@@ -448,7 +464,7 @@ static int unionfs_symlink(struct inode *dir, struct dentry *dentry,
err = PTR_ERR(hidden_dentry);

printk(KERN_DEBUG "hidden dentry NULL (or error)"
- "for bindex = %d\n", bindex);
+ "for bindex = %d\n", bindex);
continue;
}
}
@@ -458,23 +474,31 @@ static int unionfs_symlink(struct inode *dir, struct dentry *dentry,
if (!(err = is_robranch_super(dentry->d_sb, bindex))) {
mode = S_IALLUGO;
err =
- vfs_symlink(hidden_dir_dentry->d_inode,
- hidden_dentry, symname, mode);
+ vfs_symlink(hidden_dir_dentry->d_inode,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
+ hidden_dentry, symname, mode);
}
unlock_dir(hidden_dir_dentry);

if (err || !hidden_dentry->d_inode) {
- /* break out of for loop if error returned was NOT -EROFS */
+ /*
+ * break out of for loop if error returned was NOT
+ * -EROFS.
+ */
if (!IS_COPYUP_ERR(err))
break;
} else {
err = unionfs_interpose(dentry, dir->i_sb, 0);
if (!err) {
fsstack_copy_attr_times(dir,
- hidden_dir_dentry->d_inode);
+ hidden_dir_dentry->
+ d_inode);
fsstack_copy_inode_size(dir,
- hidden_dir_dentry->d_inode);
- /* update number of links on parent directory */
+ hidden_dir_dentry->
+ d_inode);
+ /*
+ * update number of links on parent
+ * directory.
+ */
dir->i_nlink = unionfs_get_nlinks(dir);
}
break;
@@ -507,7 +531,10 @@ static int unionfs_mkdir(struct inode *parent, struct dentry *dentry, int mode)

hidden_dentry = unionfs_lower_dentry(dentry);

- /* check if whiteout exists in this branch, i.e. lookup .wh.foo first */
+ /*
+ * check if whiteout exists in this branch, i.e. lookup .wh.foo
+ * first.
+ */
name = alloc_whname(dentry->d_name.name, dentry->d_name.len);
if (IS_ERR(name)) {
err = PTR_ERR(name);
@@ -515,7 +542,7 @@ static int unionfs_mkdir(struct inode *parent, struct dentry *dentry, int mode)
}

whiteout_dentry = lookup_one_len(name, hidden_dentry->d_parent,
- dentry->d_name.len + UNIONFS_WHLEN);
+ dentry->d_name.len + UNIONFS_WHLEN);
if (IS_ERR(whiteout_dentry)) {
err = PTR_ERR(whiteout_dentry);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
goto out;
@@ -559,7 +586,7 @@ static int unionfs_mkdir(struct inode *parent, struct dentry *dentry, int mode)
hidden_dentry = create_parents(parent, dentry, bindex);
if (!hidden_dentry || IS_ERR(hidden_dentry)) {
printk(KERN_DEBUG "hidden dentry NULL for "
- "bindex = %d\n", bindex);
+ "bindex = %d\n", bindex);
continue;
}
}
@@ -571,7 +598,8 @@ static int unionfs_mkdir(struct inode *parent, struct dentry *dentry, int mode)
goto out;
}

- err = vfs_mkdir(hidden_parent_dentry->d_inode, hidden_dentry, mode);
+ err = vfs_mkdir(hidden_parent_dentry->d_inode, hidden_dentry,
+ mode);

unlock_dir(hidden_parent_dentry);

@@ -590,9 +618,9 @@ static int unionfs_mkdir(struct inode *parent, struct dentry *dentry, int mode)
err = unionfs_interpose(dentry, parent->i_sb, 0);
if (!err) {
fsstack_copy_attr_times(parent,
- hidden_parent_dentry->d_inode);
+ hidden_parent_dentry->d_inode);
fsstack_copy_inode_size(parent,
- hidden_parent_dentry->d_inode);
+ hidden_parent_dentry->d_inode);

/* update number of links on parent directory */
parent->i_nlink = unionfs_get_nlinks(parent);
@@ -601,7 +629,7 @@ static int unionfs_mkdir(struct inode *parent, struct dentry *dentry, int mode)
err = make_dir_opaque(dentry, dbstart(dentry));
if (err) {
printk(KERN_ERR "mkdir: error creating "
- ".wh.__dir_opaque: %d\n", err);
+ ".wh.__dir_opaque: %d\n", err);
goto out;
}

@@ -636,7 +664,10 @@ static int unionfs_mknod(struct inode *dir, struct dentry *dentry, int mode,

hidden_dentry = unionfs_lower_dentry(dentry);

- /* check if whiteout exists in this branch, i.e. lookup .wh.foo first */
+ /*
+ * check if whiteout exists in this branch, i.e. lookup .wh.foo
+ * first.
+ */
name = alloc_whname(dentry->d_name.name, dentry->d_name.len);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
if (IS_ERR(name)) {
err = PTR_ERR(name);
@@ -644,7 +675,7 @@ static int unionfs_mknod(struct inode *dir, struct dentry *dentry, int mode,
}

whiteout_dentry = lookup_one_len(name, hidden_dentry->d_parent,
- dentry->d_name.len + UNIONFS_WHLEN);
+ dentry->d_name.len + UNIONFS_WHLEN);
if (IS_ERR(whiteout_dentry)) {
err = PTR_ERR(whiteout_dentry);
goto out;
@@ -706,9 +737,9 @@ static int unionfs_mknod(struct inode *dir, struct dentry *dentry, int mode,
err = unionfs_interpose(dentry, dir->i_sb, 0);
if (!err) {
fsstack_copy_attr_times(dir,
- hidden_parent_dentry->d_inode);
+ hidden_parent_dentry->d_inode);
fsstack_copy_inode_size(dir,
- hidden_parent_dentry->d_inode);
+ hidden_parent_dentry->d_inode);
/* update number of links on parent directory */
dir->i_nlink = unionfs_get_nlinks(dir);
}
@@ -744,9 +775,11 @@ static int unionfs_readlink(struct dentry *dentry, char __user * buf,
goto out;
}

- err = hidden_dentry->d_inode->i_op->readlink(hidden_dentry, buf, bufsiz);
+ err = hidden_dentry->d_inode->i_op->readlink(hidden_dentry,
+ buf, bufsiz);
if (err > 0)
- fsstack_copy_attr_atime(dentry->d_inode, hidden_dentry->d_inode);
+ fsstack_copy_attr_atime(dentry->d_inode,
+ hidden_dentry->d_inode);

out:
unionfs_unlock_dentry(dentry);
@@ -795,7 +828,7 @@ static void unionfs_put_link(struct dentry *dentry, struct nameidata *nd,

/* Basically copied from the kernel vfs permission(), but we've changed
* the following:
- * (1) the IS_RDONLY check is skipped, and
+ * (1) the IS_RDONLY check is skipped, and
* (2) if you set the mount option `mode=nfsro', we assume that -EACCES
* means that the export is read-only and we should check standard Unix
* permissions. This means that NFS ACL checks (or other advanced
@@ -803,8 +836,8 @@ static void unionfs_put_link(struct dentry *dentry, struct nameidata *nd,
* security_inode_permission, and therefore security inside SELinux, etc.
* are performed.
*/
-static int inode_permission(struct inode *inode, int mask, struct nameidata *nd,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
- int bindex)
+static int inode_permission(struct inode *inode, int mask,
+ struct nameidata *nd, int bindex)
{
int retval, submask;

@@ -813,7 +846,7 @@ static int inode_permission(struct inode *inode, int mask, struct nameidata *nd,
if (bindex == 0) {
umode_t mode = inode->i_mode;
if (IS_RDONLY(inode) &&
- (S_ISREG(mode) || S_ISDIR(mode) || S_ISLNK(mode)))
+ (S_ISREG(mode) || S_ISDIR(mode) || S_ISLNK(mode)))
return -EROFS;
}
/*
@@ -835,7 +868,8 @@ static int inode_permission(struct inode *inode, int mask, struct nameidata *nd,
perms = branchperms(nd->mnt->mnt_sb, bindex);
unionfs_read_unlock(nd->mnt->mnt_sb);
if (perms & MAY_NFSRO)
- retval = generic_permission(inode, submask, NULL);
+ retval = generic_permission(inode, submask,
+ NULL);
}
} else
retval = generic_permission(inode, submask, NULL);
@@ -928,7 +962,8 @@ static int unionfs_setattr(struct dentry *dentry, struct iattr *ia)
bend = dbend(dentry);
inode = dentry->d_inode;

- for (bindex = bstart; (bindex <= bend) || (bindex == bstart); bindex++) {
+ for (bindex = bstart; (bindex <= bend) || (bindex == bstart);
+ bindex++) {
hidden_dentry = unionfs_lower_dentry_idx(dentry, bindex);
if (!hidden_dentry)
continue;
@@ -950,10 +985,14 @@ static int unionfs_setattr(struct dentry *dentry, struct iattr *ia)

if (!err) {
copyup = 1;
- hidden_dentry = unionfs_lower_dentry(dentry);
+ hidden_dentry =
+ unionfs_lower_dentry(dentry);
break;
}
- /* if error is in the leftmost branch, pass it up */
+ /*
+ * if error is in the leftmost branch, pass
+ * it up.
+ */
if (i == 0)
goto out;
```

[PATCH 02/21] Unionfs: Coding style fixes

```
}
@@ -1013,4 +1052,3 @@ struct inode_operations unionfs_main_iops = {
.listxattr = unionfs_listxattr,
#endif
};
-
diff --git a/fs/unionfs/lookup.c b/fs/unionfs/lookup.c
index 5157f36..4be590f 100644
--- a/fs/unionfs/lookup.c
+++ b/fs/unionfs/lookup.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003-2007 Stony Brook University
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -46,8 +46,9 @@ static noinline int is_opaque_dir(struct dentry *dentry, int bindex)
mutex_lock(&hidden_inode->i_mutex);

if (!permission(hidden_inode, MAY_EXEC, NULL))
- wh_hidden_dentry = lookup_one_len(UNIONFS_DIR_OPAQUE, hidden_dentry,
- sizeof(UNIONFS_DIR_OPAQUE) - 1);
+ wh_hidden_dentry =
+ lookup_one_len(UNIONFS_DIR_OPAQUE, hidden_dentry,
+ sizeof(UNIONFS_DIR_OPAQUE) - 1);
else {
args.is_opaque.dentry = hidden_dentry;
run_sioq(__is_opaque_dir, &args);
@@ -69,8 +70,8 @@ out:
return err;
}

-struct dentry *unionfs_lookup_backend(struct dentry *dentry, struct nameidata *nd,
- int lookupmode)
+struct dentry *unionfs_lookup_backend(struct dentry *dentry,
+ struct nameidata *nd, int lookupmode)
{
int err = 0;
struct dentry *hidden_dentry = NULL;
@@ -153,7 +154,8 @@ struct dentry *unionfs_lookup_backend(struct dentry *dentry, struct nameidata *n
continue;
BUG_ON(hidden_dentry != NULL);

- hidden_dir_dentry = unionfs_lower_dentry_idx(parent_dentry, bindex);
+ hidden_dir_dentry =
+ unionfs_lower_dentry_idx(parent_dentry, bindex);

/* if the parent hidden dentry does not exist skip this */
```

[PATCH 02/21] Unionfs: Coding style fixes

```
if (!(hidden_dir_dentry && hidden_dir_dentry->d_inode))
@@ -208,7 +210,7 @@ struct dentry *unionfs_lookup_backend(struct dentry *dentry, struct nameidata *n
nd->mnt = unionfs_lower_mnt_idx(parent_dentry, bindex);

hidden_dentry = lookup_one_len_nd(name, hidden_dir_dentry,
- namelen, nd);
+ namelen, nd);
if (IS_ERR(hidden_dentry)) {
dput(first_hidden_dentry);
unionfs_mntput(first_dentry, first_dentry_offset);
@@ -226,7 +228,8 @@ struct dentry *unionfs_lookup_backend(struct dentry *dentry, struct nameidata *n
* to allow mountpoint crossing
*/
first_dentry = parent_dentry;
- first_hidden_mnt = unionfs_mntget(parent_dentry, bindex);
+ first_hidden_mnt =
+ unionfs_mntget(parent_dentry, bindex);
first_dentry_offset = bindex;
} else
dput(hidden_dentry);
@@ -245,12 +248,13 @@ struct dentry *unionfs_lookup_backend(struct dentry *dentry, struct nameidata *n
* mountpoint crossing
*/
unionfs_set_lower_mnt_idx(dentry, bindex,
- unionfs_mntget(parent_dentry, bindex));
+ unionfs_mntget(parent_dentry,
+ bindex));
set_dbend(dentry, bindex);

/* update parent directory's atime with the bindex */
fsstack_copy_attr_atime(parent_dentry->d_inode,
- hidden_dir_dentry->d_inode);
+ hidden_dir_dentry->d_inode);

/* We terminate file lookups here. */
if (!S_ISDIR(hidden_dentry->d_inode->i_mode)) {
@@ -259,7 +263,8 @@ struct dentry *unionfs_lookup_backend(struct dentry *dentry, struct nameidata *n
if (dentry_count == 1)
goto out_positive;
/* This can only happen with mixed D-*F-* */
- BUG_ON(!S_ISDIR(unionfs_lower_dentry(dentry)->d_inode->i_mode));
+ BUG_ON(!S_ISDIR(unionfs_lower_dentry(dentry)->
+ d_inode->i_mode));
continue;
}

@@ -298,22 +303,25 @@ out_negative:
/* FIXME: fix following line for mount point crossing */
nd->mnt = unionfs_lower_mnt_idx(parent_dentry, bindex);

- first_hidden_dentry = lookup_one_len_nd(name, hidden_dir_dentry,
```

[PATCH 02/21] Unionfs: Coding style fixes

```
- namelen, nd);
+ first_hidden_dentry =
+ lookup_one_len_nd(name, hidden_dir_dentry,
+ namelen, nd);
first_dentry_offset = bindex;
if (IS_ERR(first_hidden_dentry)) {
err = PTR_ERR(first_hidden_dentry);
goto out;
}
-
+
/* FIXME: the following line needs to be changed to allow
* mountpoint crossing
*/
first_dentry = dentry;
first_hidden_mnt = unionfs_mntget(dentry, bindex);
}
- unionfs_set_lower_dentry_idx(dentry, first_dentry_offset, first_hidden_dentry);
- unionfs_set_lower_mnt_idx(dentry, first_dentry_offset, first_hidden_mnt);
+ unionfs_set_lower_dentry_idx(dentry, first_dentry_offset,
+ first_hidden_dentry);
+ unionfs_set_lower_mnt_idx(dentry, first_dentry_offset,
+ first_hidden_mnt);
set_dbstart(dentry, first_dentry_offset);
set_dbend(dentry, first_dentry_offset);

@@ -409,9 +417,10 @@ int unionfs_partial_lookup(struct dentry *dentry)
static struct kmem_cache *unionfs_dentry_cache;
int unionfs_init_dentry_cache(void)
{
- unionfs_dentry_cache = kmem_cache_create("unionfs_dentry",
- sizeof(struct unionfs_dentry_info), 0,
- SLAB_RECLAIM_ACCOUNT, NULL, NULL);
+ unionfs_dentry_cache =
+ kmem_cache_create("unionfs_dentry",
+ sizeof(struct unionfs_dentry_info),
+ 0, SLAB_RECLAIM_ACCOUNT, NULL, NULL);

return (unionfs_dentry_cache ? 0 : -ENOMEM);
}
@@ -440,7 +449,7 @@ int new_dentry_private_data(struct dentry *dentry)
spin_lock(&dentry->d_lock);
if (!info) {
dentry->d_fsdata = kmem_cache_alloc(unionfs_dentry_cache,
- GFP_ATOMIC);
+ GFP_ATOMIC);
info = UNIONFS_D(dentry);

if (!info)
@@ -514,4 +523,3 @@ void update_bstart(struct dentry *dentry)
unionfs_set_lower_dentry_idx(dentry, bindex, NULL);
```

[PATCH 02/21] Unionfs: Coding style fixes

```
}
}
-
diff --git a/fs/unionfs/main.c b/fs/unionfs/main.c
index 4fffafa..ffedbcd 100644
--- a/fs/unionfs/main.c
+++ b/fs/unionfs/main.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003–2007 Stony Brook University
- * Copyright (c) 2003–2007 The Research Foundation of State University of New York
+ * Copyright (c) 2003–2007 The Research Foundation of SUNY
*
* This program is free software; you can redistribute it and/or modify
* it under the terms of the GNU General Public License version 2 as
@@ -61,7 +61,7 @@ int unionfs_interpose(struct dentry *dentry, struct super_block *sb, int flag)
atomic_read(&UNIONFS_SB(sb)->generation));

UNIONFS_I(inode)->lower_inodes =
- kcalloc(sbmax(sb), sizeof(struct inode *), GFP_KERNEL);
+ kcalloc(sbmax(sb), sizeof(struct inode *), GFP_KERNEL);
if (!UNIONFS_I(inode)->lower_inodes) {
err = -ENOMEM;
goto out;
@@ -90,7 +90,7 @@ int unionfs_interpose(struct dentry *dentry, struct super_block *sb, int flag)
continue;

unionfs_set_lower_inode_idx(inode, bindex,
- igrab(hidden_dentry->d_inode));
+ igrab(hidden_dentry->d_inode));
}

ibstart(inode) = dbstart(dentry);
@@ -169,7 +169,7 @@ void unionfs_reinterpose(struct dentry *dentry)
if (unionfs_lower_inode_idx(inode, bindex))
continue;
unionfs_set_lower_inode_idx(inode, bindex,
- igrab(hidden_dentry->d_inode));
+ igrab(hidden_dentry->d_inode));
}
ibstart(inode) = dbstart(dentry);
ibend(inode) = dbend(dentry);
@@ -232,6 +232,7 @@ int __parse_branch_mode(const char *name)
int parse_branch_mode(const char *name)
{
int perms = __parse_branch_mode(name);
+
if (perms == 0)
perms = MAY_READ | MAY_WRITE;
return perms;
}
```

[PATCH 02/21] Unionfs: Coding style fixes

```
@@ -266,15 +267,15 @@ static int parse_dirs_option(struct super_block *sb, struct unionfs_dentry_info
branches++;
```

```
/* allocate space for underlying pointers to hidden dentry */
- UNIONFS_SB(sb)->data = kcalloc(branches,
- sizeof(struct unionfs_data), GFP_KERNEL);
+ UNIONFS_SB(sb)->data =
+ kcalloc(branches, sizeof(struct unionfs_data), GFP_KERNEL);
if (!UNIONFS_SB(sb)->data) {
err = -ENOMEM;
goto out;
}
```

```
- hidden_root_info->lower_paths = kcalloc(branches,
- sizeof(struct path), GFP_KERNEL);
```

```
+ hidden_root_info--
```

```
diff --git a/fs/unionfs/rdstate.c b/fs/unionfs/rdstate.c
```

```
index b67a86a..4e875b1 100644
```

```
--- a/fs/unionfs/rdstate.c
```

```
+++ b/fs/unionfs/rdstate.c
```

```
@@ -9,7 +9,7 @@
```

```
* Copyright (c) 2003 Puja Gupta
```

```
* Copyright (c) 2003 Harikesavan Krishnan
```

```
* Copyright (c) 2003-2007 Stony Brook University
```

```
- * Copyright (c) 2003-2007 The Research Foundation of State University of New York
```

```
+ * Copyright (c) 2003-2007 The Research Foundation of SUNY
```

```
*
```

```
* This program is free software; you can redistribute it and/or modify
```

```
* it under the terms of the GNU General Public License version 2 as
```

```
@@ -34,8 +34,9 @@ static struct kmem_cache *unionfs_filldir_cachep;
```

```
int unionfs_init_filldir_cache(void)
```

```
{
```

```
unionfs_filldir_cachep =
```

```
- kmem_cache_create("unionfs_filldir", sizeof(struct filldir_node), 0,
```

```
- SLAB_RECLAIM_ACCOUNT, NULL, NULL);
```

```
+ kmem_cache_create("unionfs_filldir",
```

```
+ sizeof(struct filldir_node), 0,
```

```
+ SLAB_RECLAIM_ACCOUNT, NULL, NULL);
```

```
return (unionfs_filldir_cachep ? 0 : -ENOMEM);
```

```
}
```

```
@@ -74,7 +75,8 @@ static int guesstimate_hash_size(struct inode *inode)
```

```
if (hidden_inode->i_size == DENTPAGE)
```

```
hashsize += DENTPERONEPAGE;
```

```
else
```

```
- hashsize += (hidden_inode->i_size / DENTPAGE) * DENTPERPAGE;
```

```
+ hashsize += (hidden_inode->i_size / DENTPAGE) *
```

```
+ DENTPERPAGE;
```

```
}
```

```
return hashsize;
```

[PATCH 02/21] Unionfs: Coding style fixes

```
@@ -82,12 +84,13 @@ static int guesstimate_hash_size(struct inode *inode)

int init_rdsta(struct file *file)
{
- BUG_ON(sizeof(loff_t) != (sizeof(unsigned int) + sizeof(unsigned int)));
+ BUG_ON(sizeof(loff_t) !=
+ (sizeof(unsigned int) + sizeof(unsigned int)));
BUG_ON(UNIONFS_F(file)->rdstate != NULL);

UNIONFS_F(file)->rdstate = alloc_rdsta(file->f_dentry->d_inode,
fbstart(file));
-
+
return (UNIONFS_F(file)->rdstate ? 0 : -ENOMEM);
}

@@ -99,7 +102,7 @@ struct unionfs_dir_state *find_rdsta(struct inode *inode, loff_t fpos)
spin_lock(&UNIONFS_I(inode)->rdlock);
list_for_each(pos, &UNIONFS_I(inode)->readdircache) {
struct unionfs_dir_state *r =
- list_entry(pos, struct unionfs_dir_state, cache);
+ list_entry(pos, struct unionfs_dir_state, cache);
if (fpos == rdstate2offset(r)) {
UNIONFS_I(inode)->rdcount--;
list_del(&r->cache);
@@ -126,8 +129,8 @@ struct unionfs_dir_state *alloc_rdsta(struct inode *inode, int bindex)
if (mallosize > PAGE_SIZE)
mallosize = PAGE_SIZE;

- hashsize = (mallosize -
- sizeof(struct unionfs_dir_state)) / sizeof(struct list_head);
+ hashsize = (mallosize - sizeof(struct unionfs_dir_state)) /
+ sizeof(struct list_head);

rdstate = kmalloc(mallosize, GFP_KERNEL);
if (!rdstate)
@@ -211,9 +214,9 @@ struct filldir_node *find_filldir_node(struct unionfs_dir_state *rdstate,
*/
if (cursor->bindex == rdstate->bindex) {
printk(KERN_DEBUG "Possible I/O error "
- "unionfs_filldir: a file is duplicated "
- "in the same branch %d: %s\n",
- rdstate->bindex, cursor->name);
+ "unionfs_filldir: a file is duplicated "
+ "in the same branch %d: %s\n",
+ rdstate->bindex, cursor->name);
}
break;
}
@@ -272,4 +275,3 @@ int add_filldir_node(struct unionfs_dir_state *rdstate, const char *name,
out:
```

[PATCH 02/21] Unionfs: Coding style fixes

```
return err;
}
-
diff --git a/fs/unionfs/rename.c b/fs/unionfs/rename.c
index 0044492..5c6a33f 100644
--- a/fs/unionfs/rename.c
+++ b/fs/unionfs/rename.c
@@ -9,7 +9,7 @@
* Copyright (c) 2003 Puja Gupta
* Copyright (c) 2003 Harikesavan Krishnan
* Copyright (c) 2003–2007 Stony Brook University
- * Copyright (c) 2003–2007 The Research F
```