Union Mount

VFS based File System Namespace Unification for Linux

Bharata B Rao
<bharata@linux.vnet.ibm.com>

IBM Linux Technology Center

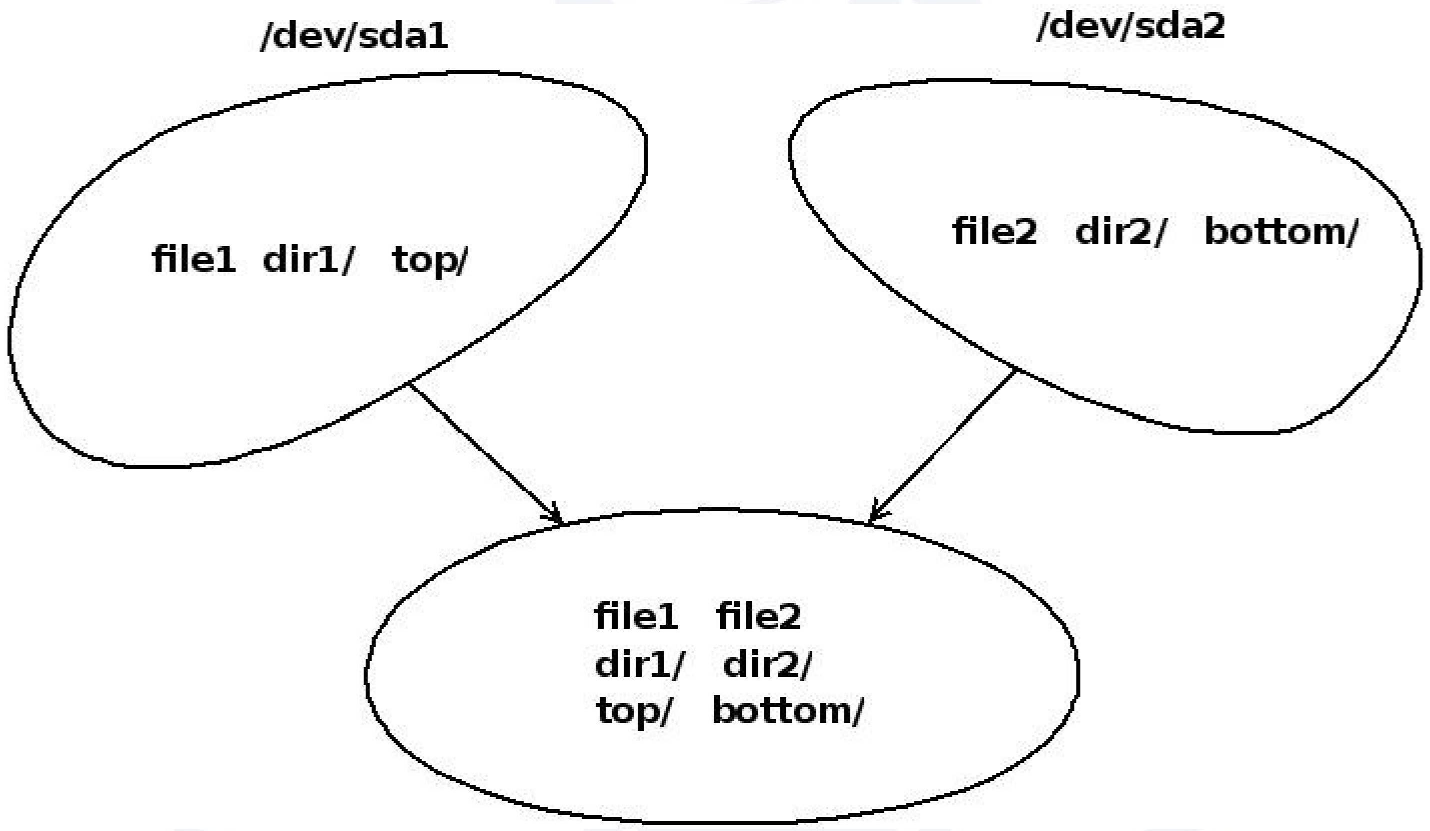
December 2007

Agenda

- Introduction to Union Filesystems.
- Basics of Filesystem Mounting.
- Union Mount.
- Semantics.
- Internals of Union Mount.
- Comparison with FS based approaches.
- Getting involved.

File System Namespace Unification

- Concept of **merging** the contents of two or more directories/filesystems to present a unified view.



Users of Unification

- **Live CD systems** – Writable RAM based FS combined with a read only FS on CD, thus allowing a writable disk-less system.
- **Server Consolidation** – Many servers sharing a common RO installation.
- **Disk-less NFS-root clients** – Set of machines sharing a single RO NFS root filesystem.
- **Sandboxing**
  - Simulation of software updates.
  - Testing OS updates.

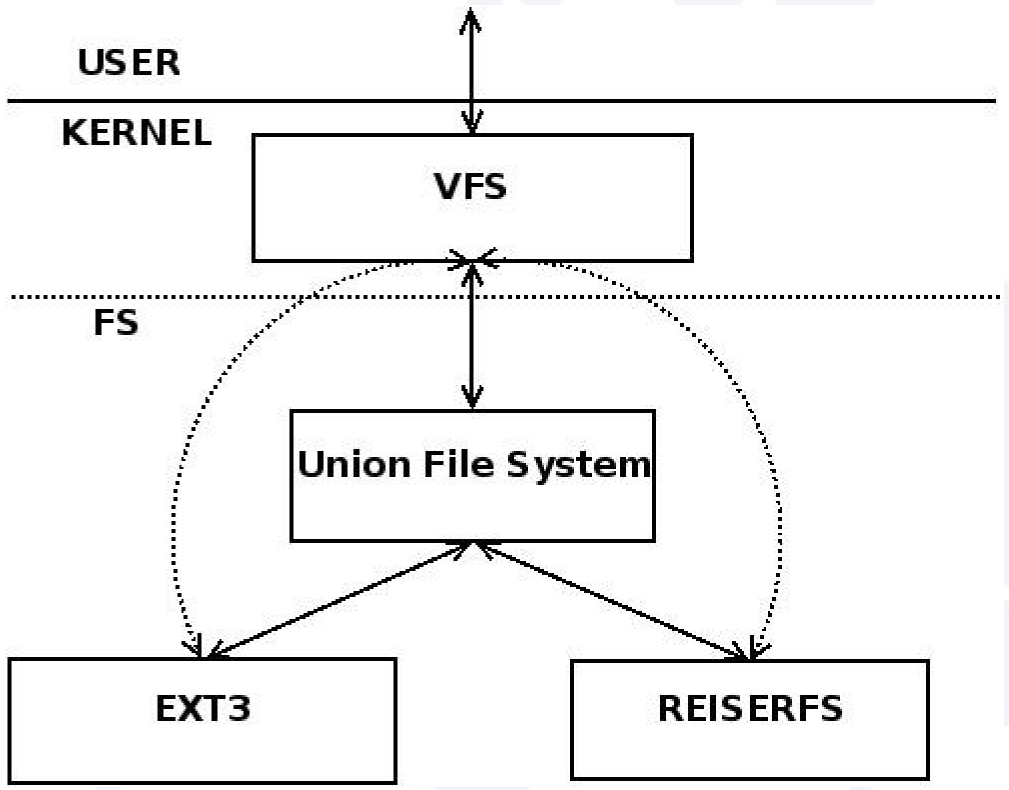
History

- *Sun's Translucent Filesystem (TLS)* provided filesystem unification.
- *BSD* provided a fully featured union mount implementation with whiteout and copyup support.
- *Plan 9* had Union Directories.
- *MAC OS X* inherited union filesystems from BSD, but didn't provide whiteout support.

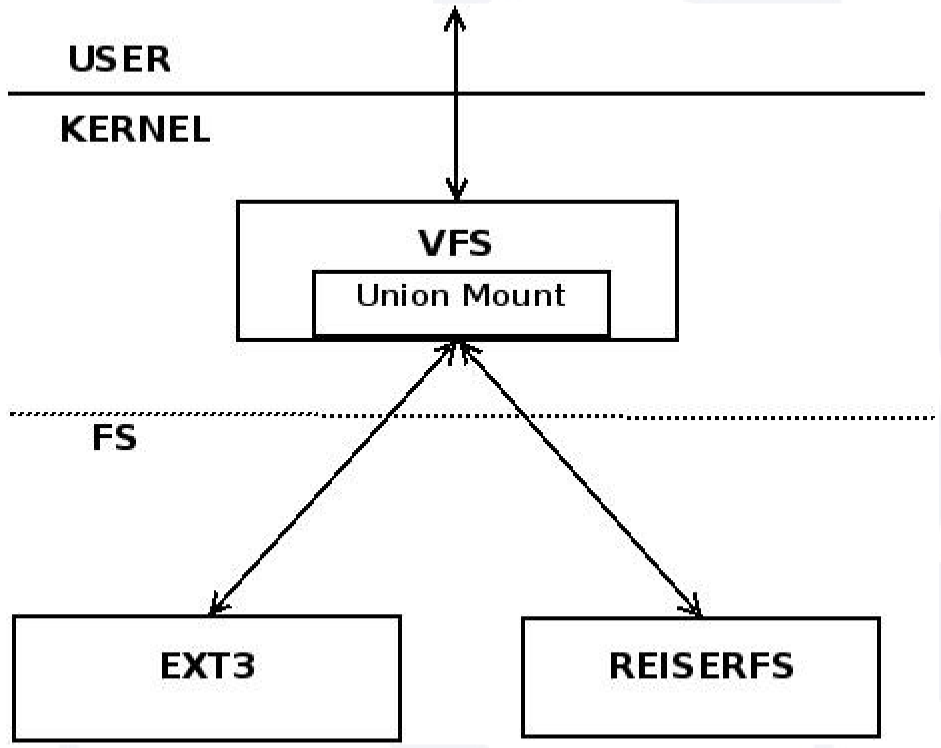
Linux Approaches

- **Unionfs** (in -mm) from Stony Brook University.
- **Aufs** a fork of Unionfs.
- Both Unionfs and Aufs are filesystem based approaches.
- **Union Mount** is a Virtual File System (VFS) based approach to filesystem namespace unification. (Original patches by Jan Blunck)

Unification at FS Layer

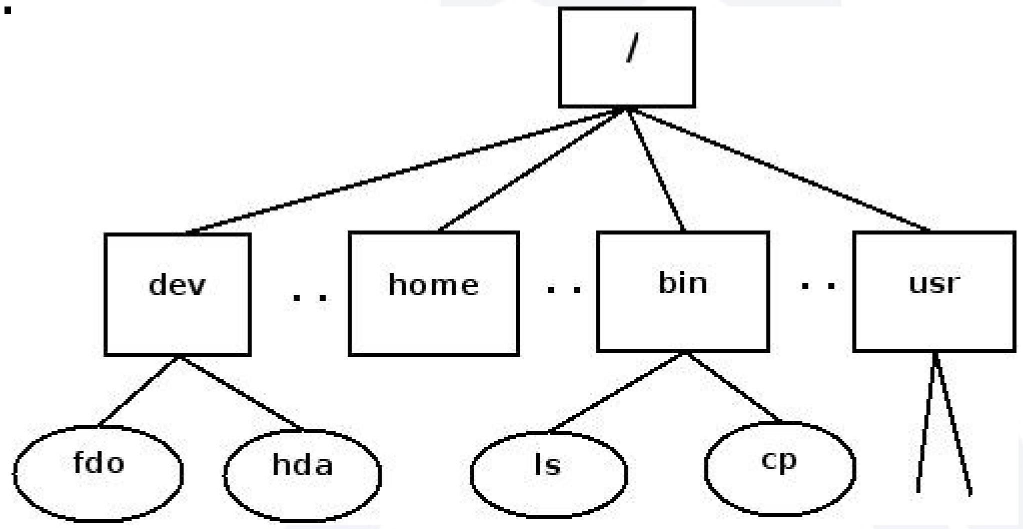


Unification at VFS Layer



File System Mounting

- **Namespace** – A hierarchical view of the filesystem contents.



- **Mounting** – Adding the FS in the device to the namespace tree.
- Eg: `mount -t ext3 /dev/sda1 /mnt`
  - `ext3` – FS type, `/dev/sda1` – Device, `/mnt` – Mount point
- `struct path { struct vfsmount *, struct dentry * };` uniquiely identifes a file in a namespace across the system

Union Mount

- Transparent mounts
  - `mount /dev/sda1 /mnt`
  - `mount –union /dev/sda2 /mnt`
- /mnt becomes the union mount point of sda1 and sda2.
- sda2 becomes the topmost writable layer.
- sda1 is the RO bottom layer of the union.

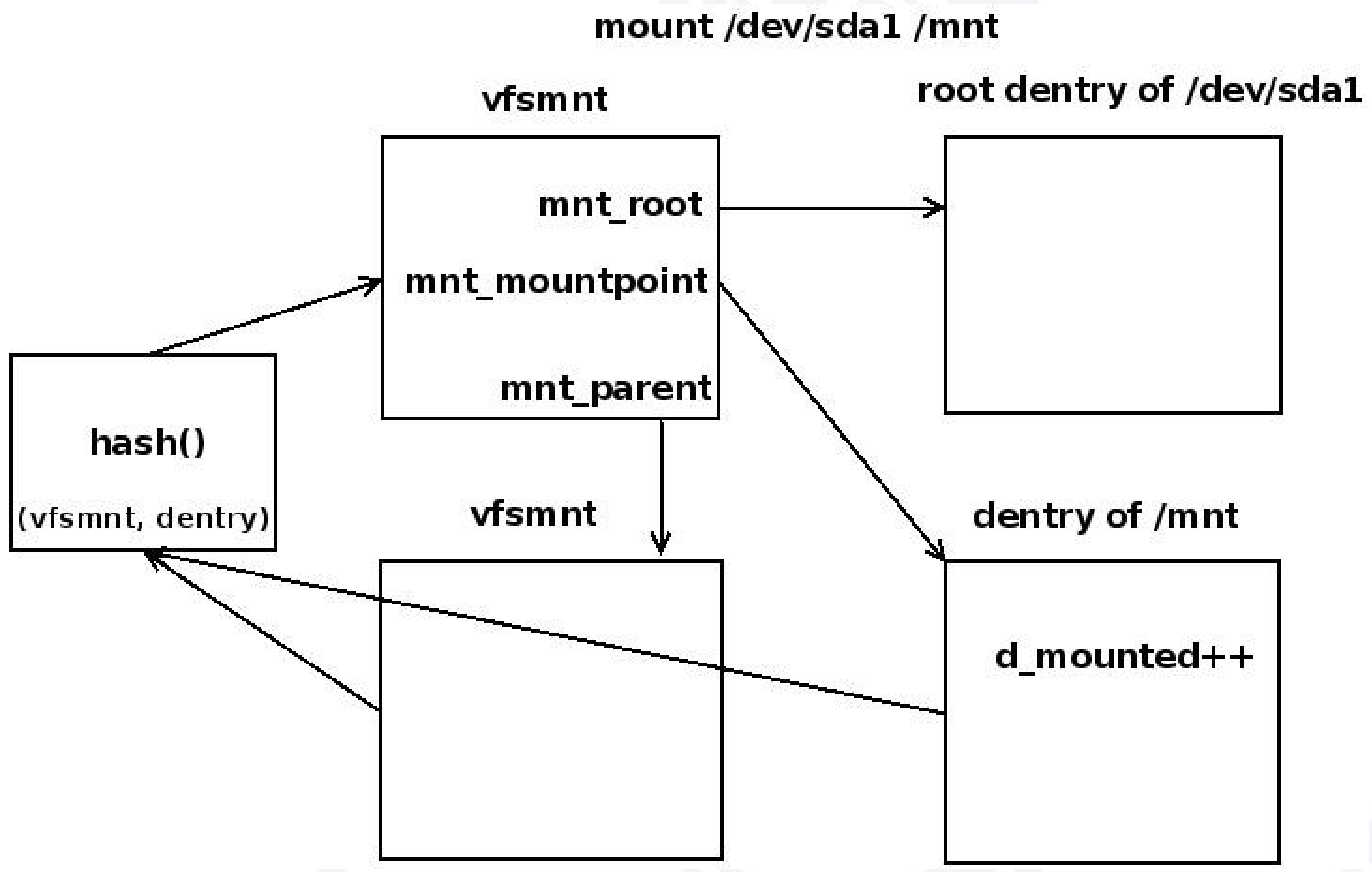
Union Mount Semantics

- **Directory listing (readdir)**
  - Merged contents of all directories of union.
  - For same named files in multiple layers, only top layer file is shown.
  - Same named directories are merged again.
- **Lookup**
  - Starts with topmost directory and proceeds downwards.
  - Stops and returns when the required file is found.
  - Descends into all lower layers in case of directories to create **subdirectory level unions.**

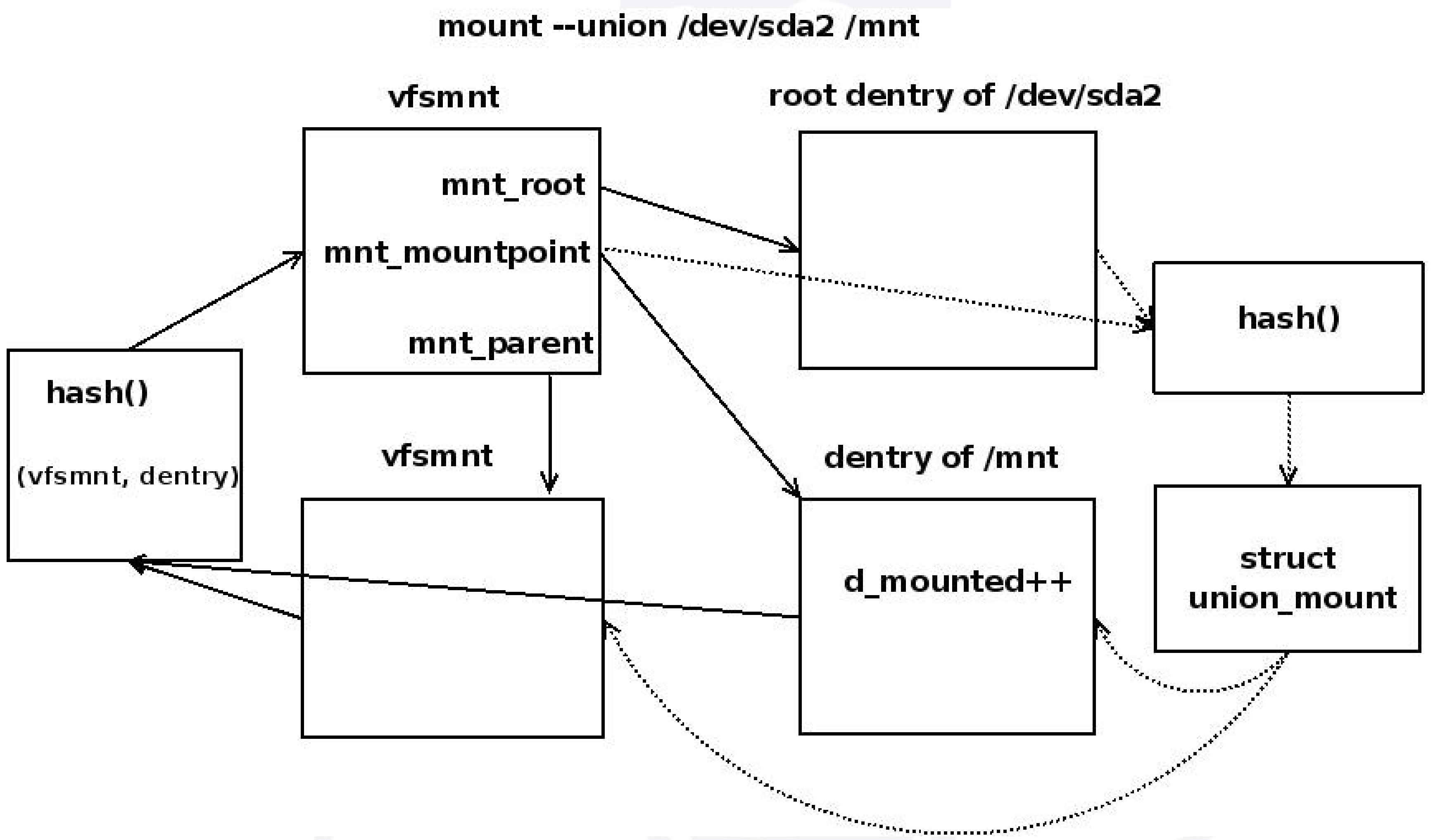
... Union Mount Semantics

- **RO lower layers, copyup**
  - All but the topmost layer are RO immutable layers.
  - Write to a lower layer file results in the file getting copied to topmost layer and write being performed on the copy.
  - Creates **shadow directories** if needed during copyup.
- **Whiteouts**
  - Place holders for files that don't exist logically.
  - Deletion of a lower level only file/directory creates a whiteout for it in the topmost directory.
  - Whiteout lookup returns -ENOENT.

Normal Mount



Union Mount


mount --union /dev/sda2 /mnt
vfsmnt
root dentry of /dev/sda2
mnt_root
mnt_mountpoint
mnt_parent
hash()
hash()
(vfsmnt, dentry)
vfsmnt
dentry of /mnt
d_mounted++
struct union_mount

Union Stack

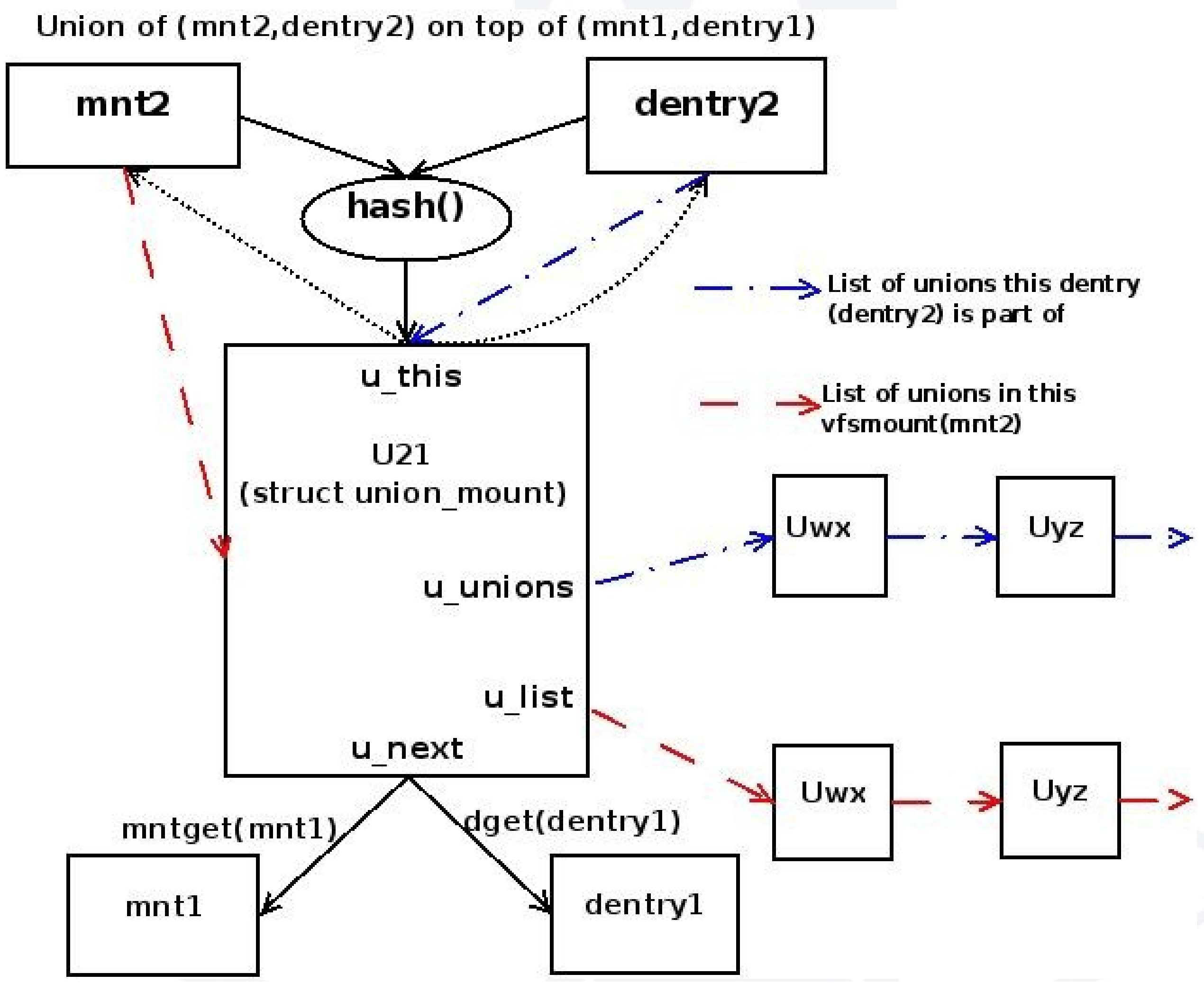
- Different layers of union are maintained as stack in VFS.
- (vfsmount, dentry) pairs are used as building blocks of union stack.
- Two layers of a union are linked together using a **union_mount** structure.

```
struct union_mount {
   struct list_head u_unions;
   struct list_head u_list;
   struct hlist_node u_hash;
   struct hlist_node u_rhash;
   struct path u_this;
   struct path u_next;
};
```

… Union Stack

- Union stacks are **built** from two places:
  - During mount operation or mount propagation.
  - During lookup of a directory that is present in more than one layer of the union.
- Union stacks are **destroyed** from two places:
  - During an un-mount operation.
  - When the upper layer dentry is destroyed after it becomes unused.

Union Mount Structure


Union of (mnt2,dentry2) on top of (mnt1,dentry1)
mnt2
dentry2
hash()
List of unions this dentry (dentry2) is part of
List of unions in this vfsmount(mnt2)
u_this
U21
(struct union_mount)
u_unions
u_list
u_next
Uwx
Uyz
Uwx
Uyz
mntget(mnt1)
dget(dentry1)
mnt1
dentry1

Directory Listing

- Directory entries are read using **getdents(2)** or **readdir(2).**

```
struct dirent {
   long d_ino;   /* inode number */
   off_t d_off;   /* offset to this dirent */
   unsigned short d_reclen; /*length of d_name*/
   char d_name[NAME_MAX+1]; /* filename */
};
```

- Dirents are stored in a cache as and when they are read.
- Dirents from all but the topmost layer are compared against this cache to eliminate duplication.
- **TODO:**
  - An approach which works for all filesystems.
  - An approach which supports llseek(2).

Copyup

- Write to a lower layer file is performed after copying the file to the topmost layer.
- **Copy on Open:** Copy is done when the file is opened for writing.
- Lookup path has been modified to create the shadow directories in the topmost layer.
- In-kernel file to file copy using **splice**.
- **TODO:**
  - Currently only copyup of regular files supported.
  - Support copyup from other places like chmod(2).
  - Need to handle links correctly.

Whiteout

- Whiteouts are necessary to provide writable unions.
- Whiteouts are handled entirely within kernel and they are transparent to users.
- Added **whiteout()** inode operation.
- Filesystems need to implement ->whiteout() to provide whiteout support.
- Typically filesystems are expected to create and use a singleton whiteout inode for all whiteout files in the filesystem.
- **TODO**:
  - Whiteout support available only for tmpfs, ext2/3/4 and need to add support for other filesystems.

Rename

- For files and directories present only in the topmost layer, traditional rename is used.
- Rename of a directory which is part of a union or which is present only in the lower layer is deferred to userspace by returning -EXDEV.
- Renaming of a regular file present only in the lower layer is done by copying it up to the topmost layer.
- For both source and target of rename, shadow directories are appropriately created during rename.

Problems with FS based approaches

- Stack information maintained by a separate filesystem.
- Pseudo VFS objects (like dentry, inode, file) maintained which link to real VFS objects from the underlying filesystems.
- Maintaining coherency between union filesystem and the underlying filesystem needs extra efforts.
  - Direct additions/deletions.
  - Direct modifications: metadata and page cache coherency.

Opportunities for Contributions(as of Nov 2007)

- Union Mount is still a work in progress and patches are in RFC state.
- Not much consensus has been reached on many aspects (Eg. directory listing), so **there is a scope to get involved and contribute.**
- Patches are mostly not tested thoroughly and there exists some corner cases where it breaks.
- **Writing Union Mount test cases for LTP is highly desired.**

Union Mount Patches

- Union Mount doesn't have a project site of its own and most development, postings happen on linux-kernel and linux-fsdevel mailing lists.
- Our last posting: http://lkml.org/lkml/2007/7/30/193
- Recent patches can be found at: ftp://ftp.suse.com/pub/people/jblunck/patches/ (temporary)
- Needs changes to util-linux package to support –*union* mount option.

Legal Statement