The Design and Implementation of a Recovery Mechanism of Deleted Files for the Linux Ext2 File System

Samir Raizada
Department of Computer Science
University of Kentucky

The Design and Implementation of a Recovery Mechanism of Deleted Files for the Linux Ext2 File System

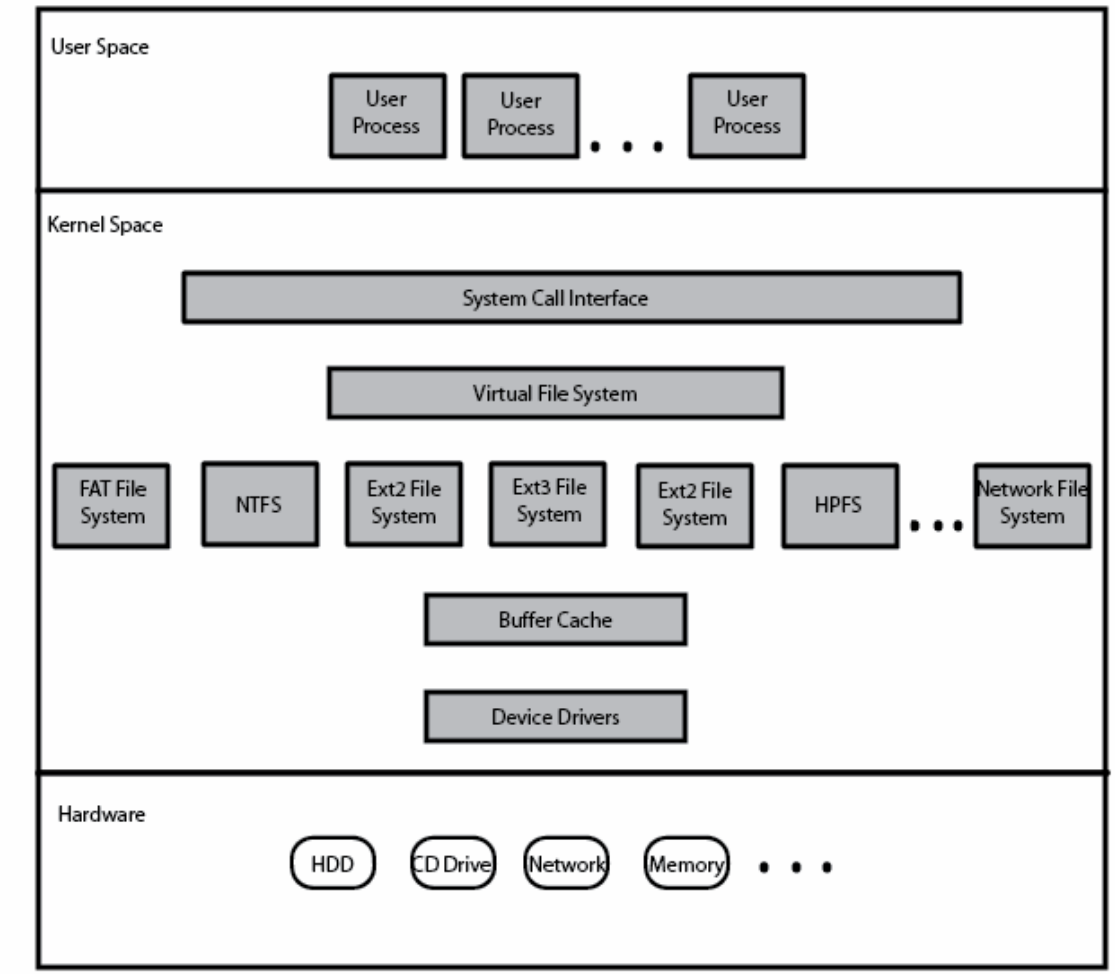
- Problem

  To provide a way for a user to recover deleted files on the Linux Ext2 file system.

The Design and Implementation of a Recovery Mechanism of Deleted Files for the Linux Ext2 File System

- Outline
  - Linux File System Framework
    - Second Extended File System
    - File Operations
  - Undelete
    - Possible Implementations
    - Implemented Approach
    - Future Work

Linux File System Framework



Linux File System Framework

- File System Objects
  - Superblock object
  - Inode object
  - File object
  - Dentry object

File System Operations

- Register File System
  - register_filesystem
    - Adds file system to kernel
    - Links the file_system_type into kernel structures
  - unregister_filesystem
    - Remove file system
    - Unlinks the file_system_type

Inode Operations

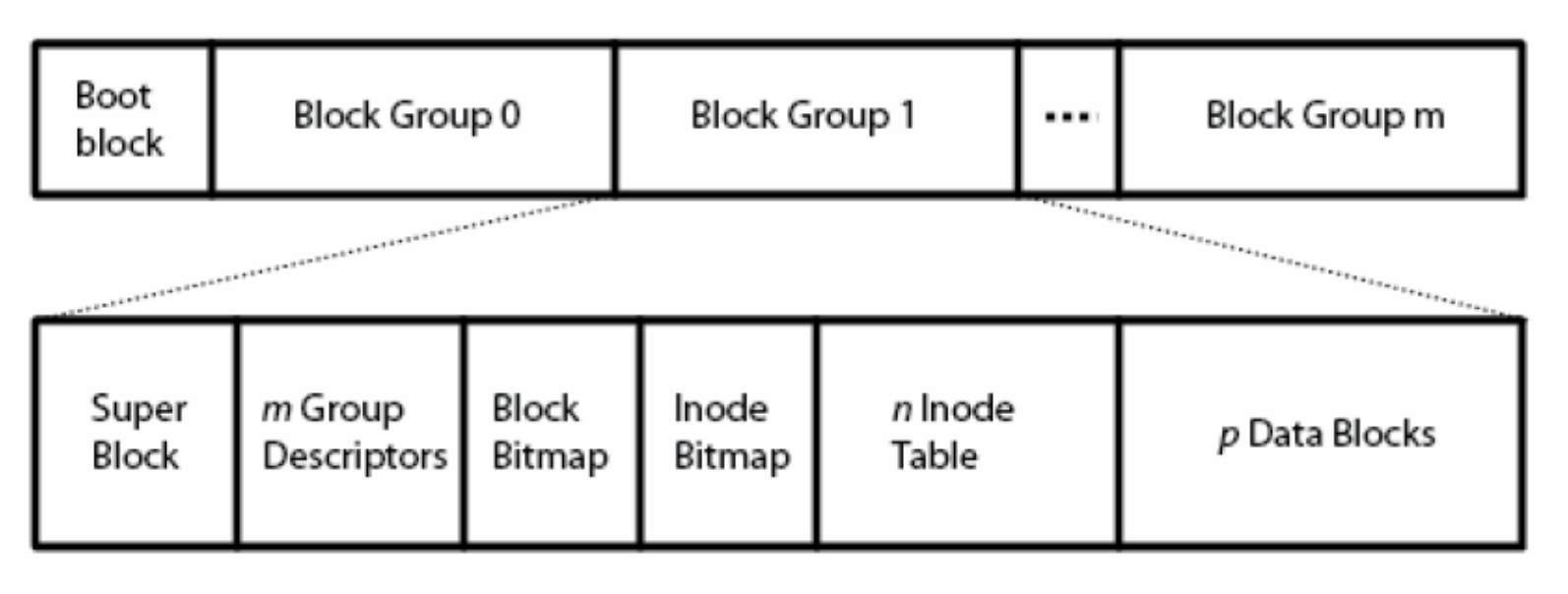
- Basic Inode operations:
  - create
  - lookup
  - link
  - **unlink**
  - mkdir
  - rmdir
  - mknod
  - readlink
  - follow_link

Ext2 File System

- Improvement on Extended File system
- Some features:
  - Partition size- 4TB
  - Longer file names
  - Block size
  - Fast symbolic links
  - File system state
  - Synchronous updates
  - Secure deletion.

Ext2 File System

- Physical Layout of the partition



Relevant Operations

- unlink
  - deletes a reference to a file corresponding to a filename from a directory
  - Inodes and data blocks freed when reference count = 0
  - VFS layer
    - resolves name
    - checks permission
    - calls filesystem’s unlink
  - Ext2 layer
    - invalidates the directory entry
    - decrements inode reference count.

Relevant operations

- Free Inode
  - Ext2 layer
    - Frees buffers
    - Marks as free in inode bitmap
    - updates free counts
- Free Data Blocks
  - Ext2 layer
    - Marks block as free in block bitmap
    - Updates free counts

rm

- used to delete one or more files
- part of the **coreutils** package
  - Uses glibc
- Operation:
  - Load libraries
  - Calls lstat
  - Verifies permissions.
  - unlink system call

Possible implementations of Undelete

- **Trash folder**
- **Delayed deletion**
- **Recovering inodes**
- **Support at the VFS layer**

Possible implementations

- Trash Folder
  - Transferring to trash folder
  - lacks transparency
  - require manual deletion
  - Implemented in different versions of Microsoft Windows Operating Systems
  - May or may not be file system specific.

Possible Implementations

- Recovering Inodes
  - No modifications to unlink
  - May store some data related to file
  - Transparent
  - Space not counted in quota
  - Can’t guarantee recovery
  - Specific to specific file system

Possible Implementations

- **Support at the VFS layer**
  - Storing the deleted file information
  - Actually delete files at a later time
  - But, directory entries are maintained by the specific file systems.
    - Essential since files may still be present
    - So it becomes dependent on specific file system implementation
    - One solution: Filter out the results of getdents call.

Possible implementations

- Delayed Deletion
  - Preserve the inode and data blocks for a predefined amount of time
  - Two possible methods
    - Initial Reference Count
      - Start with reference count of 1 (instead of 0)
    - Delayed decrementing of reference count

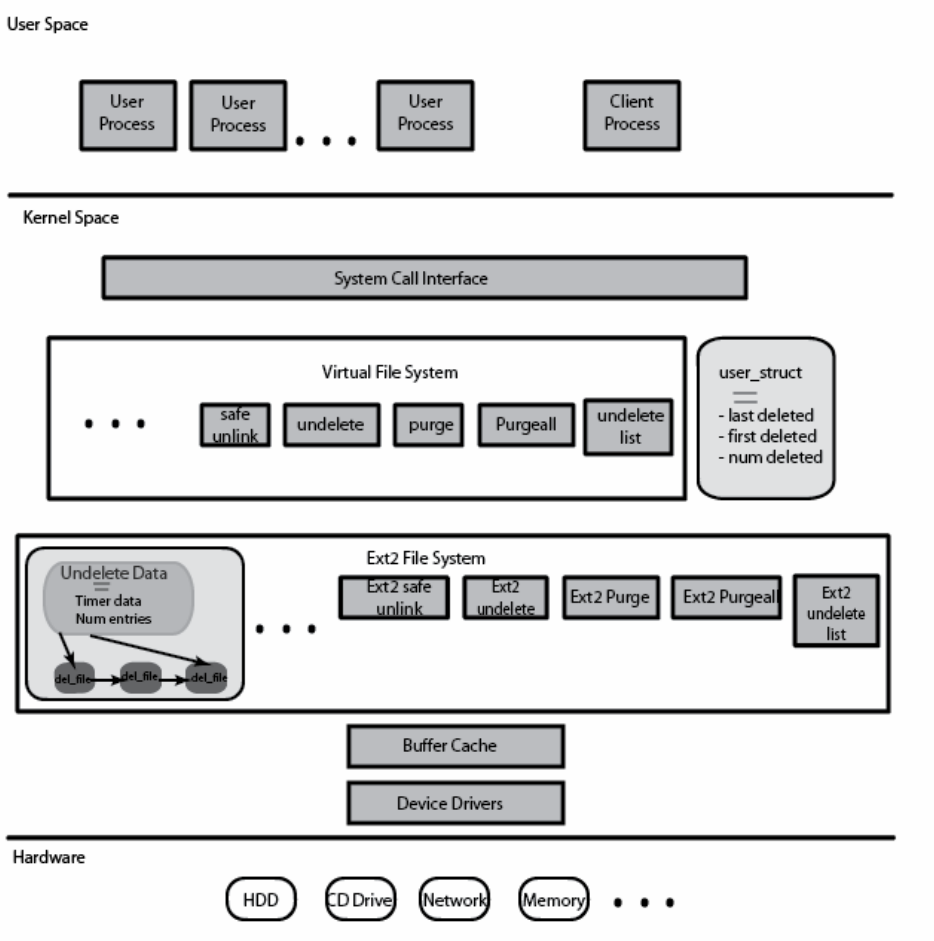
Implementated Approach

- Delayed decrementing of the inode reference count
- Directory entry invalidated
- Inode reference count remains same
- Reference count decremented after a predefined time.
- Recovery by file name
- Purge operation to get rid of the file for quota reasons.

Implementation details

- Maintains:
  - user ID of owner, file name and inode information
  - Newly deleted entry goes at the tail
  - Search from head
- Timer to purge

Implementation details



New System Calls

- safeunlink
- undelete_list
- undelete
- purge
- purgeall

safeunlink

- Modification of unlink
- Puts a new entry at the tail of `udata` structure
- Deletes directory entry (at ext2 layer)
- Starts timer used to purge files

undelete_list

- Retrieve list of files available for recovery
- Traverse linked-list (in `udata` structure)
- Return absolute path names of recoverable files
- All the files on this list may not be recoverable

undelete

- Recovers a deleted file
- Absolute file name as parameter
- Traverses the linked-list
- Searches based on
  - Owner’s user-ID
  - File name

purge and purgeall

- Purges the data stored for single or multiple deleted file
- Decrements reference count (but the inode and data blocks may not be freed)
- Searches for the file name in `udata` and deletes the node.

Kernel Files Modified

- arch/i386/kernel/entry.S
- include/asm-i386/unistd.h
- include/linux/fs.h
- include/linux/syscalls.h
- fs/Kconfig
- fs/namei.c
- fs/ext2/namei.c

Utilities

- Undelete utility
  - A simple utility to recover file
  - Uses implemented system calls to support list of recoverable files, undelete and purge operations.
- saferm
  - Implemented a simple utility to replace rm shell command
  - Uses the safeunlink system call

Status

- There is a crash in sync operation
- Still working on it

Future work

- Possibility of using EXT2_UNRM_FL file attribute.
- May support directory structure.
- May use persistent storage to store `udata` structure
- Extend the approach to other file system implementations.
- One can implement the VFS file system implementation.
- A system call to set the recoverable time period (per user).

References

- Linux Source Code (Kernel 2.6.5).
- Daniel Bovet and Marco Caseti, Understanding the Linux Kernel, Second and Third Edition, O'Reilly.
- Uresh Vahalia, Unix Internals: The new frontiers, Pearson Education.2001.
- Michael Beck et. al., Linux Kernel Internals, Second Edition, Addison Wesley 2000.
- Design and Implementation of the Second Extended Filesystem,
- Jonathan Corbet and Alessandro Rubini, Linux Device Drivers, Third Edition, O'Reilly.
- Gerlof Langeveld, Linux Kernel Internals: The File Subsystem, AT computing, Apr 2003.

Questions

THANKS