Lecture 37 - File Systems: Shared Files, Performance

CprE 308

April 14, 2014

Intro

Todays topics: File System Implementation

- Shared Files
- Buffer Cache and File System Consistency

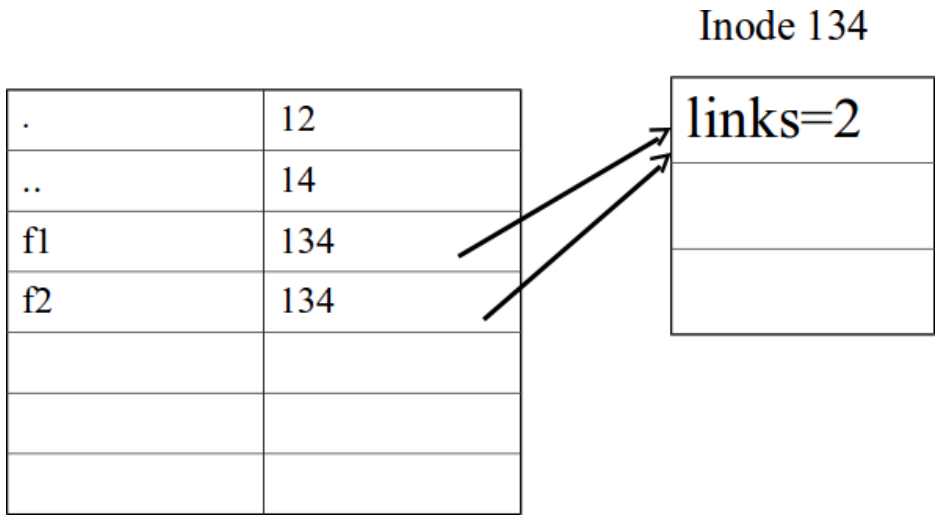
Sharing

Sharing of Files

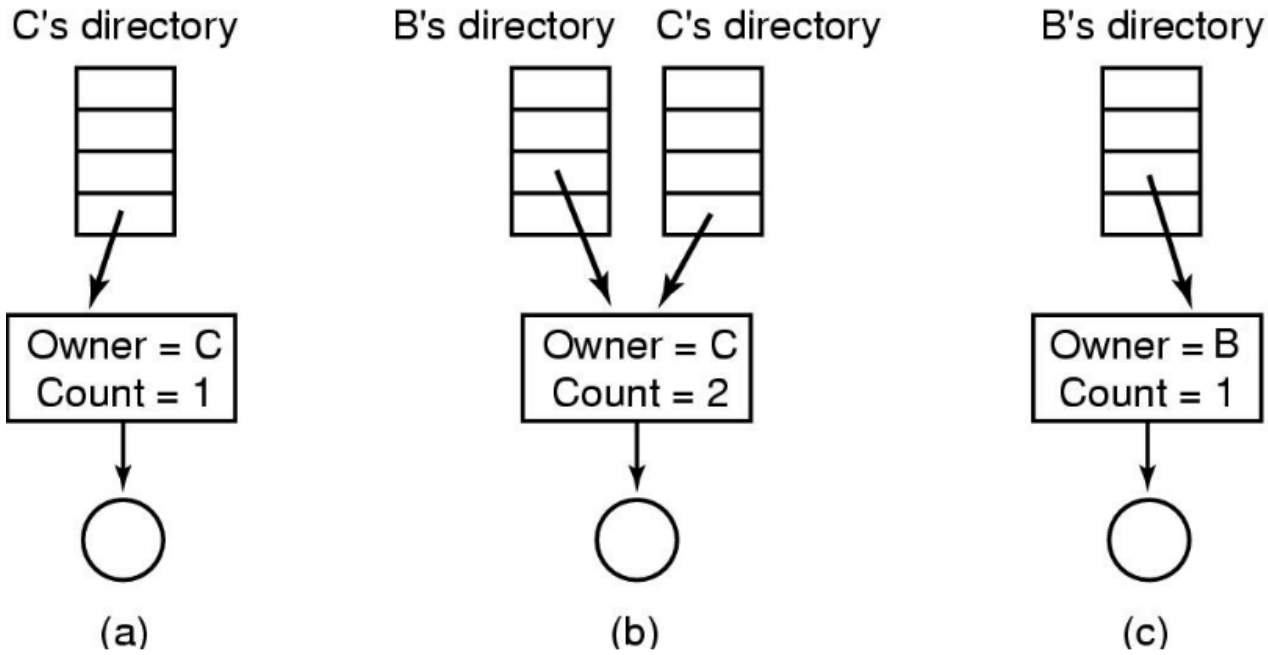
- In UNIX: `ln src dest`
- Two ways of linking files
  - "hard" links
  - Symbolic links

Hard links

- Both files point to the same inode
- `ln /home/guan/f1 /home/guan/f2`



Hard Links



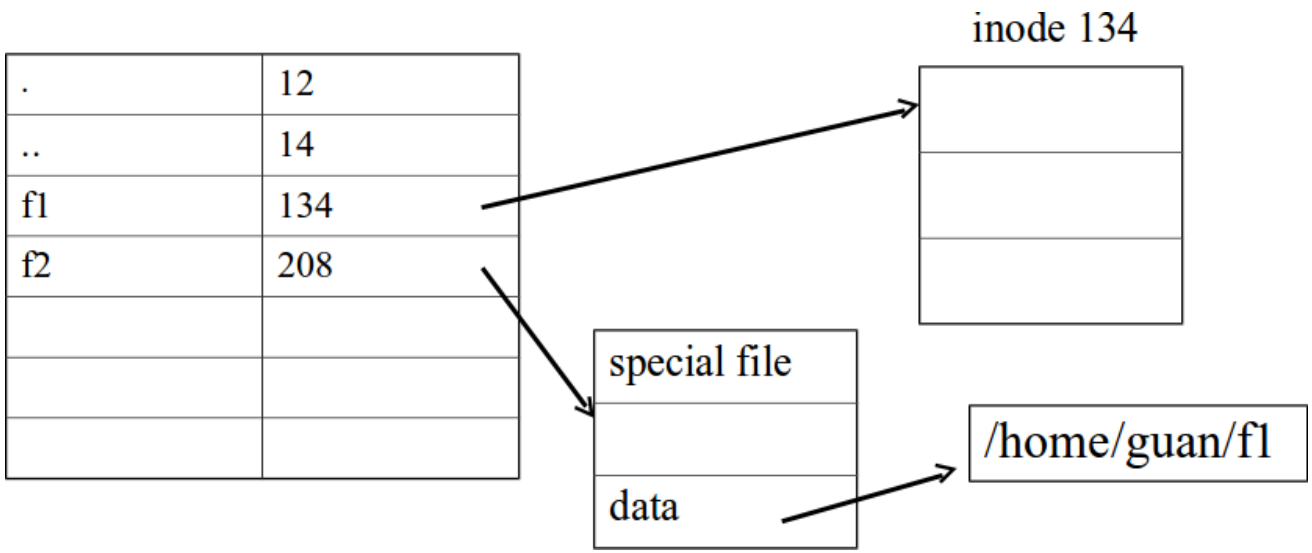
(a) Situation prior to linking

(b) After the link is created

(c) After the original owner removes the file

Symbolic links

- Files point to different inodes
- `ln -s /home/guan/f1 /home/guan/f2`

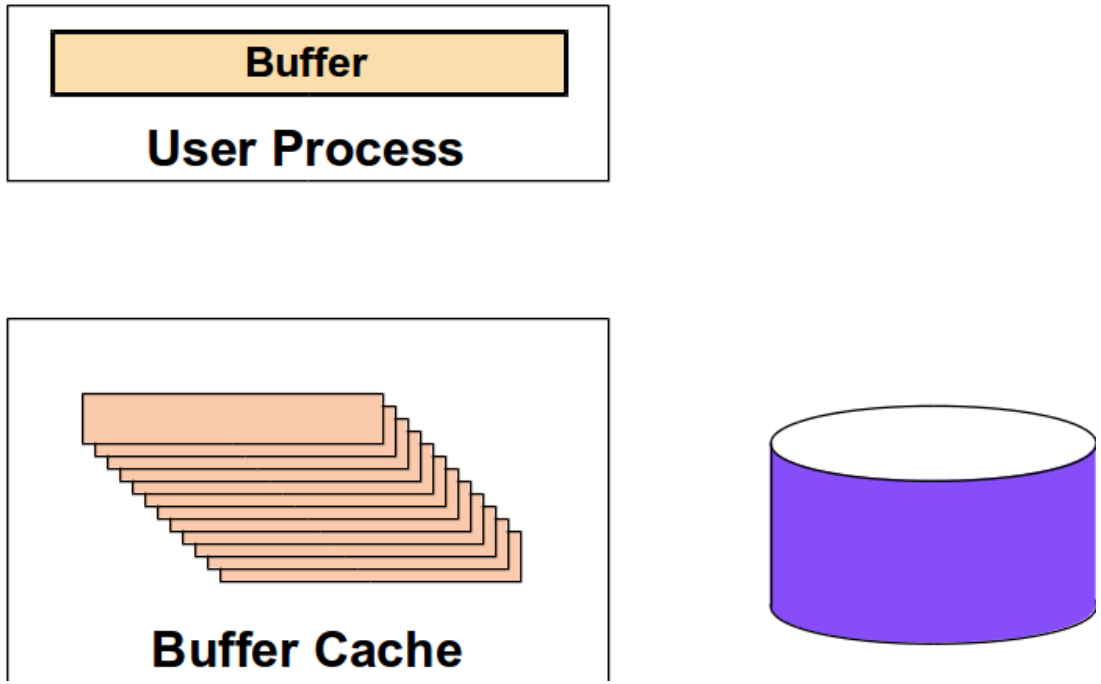


Performance

Performance of File System

- Where does your data go after a `write()` system call?
- Where does the data come from for a `read()`?
- Think about performance

The Buffer Cache

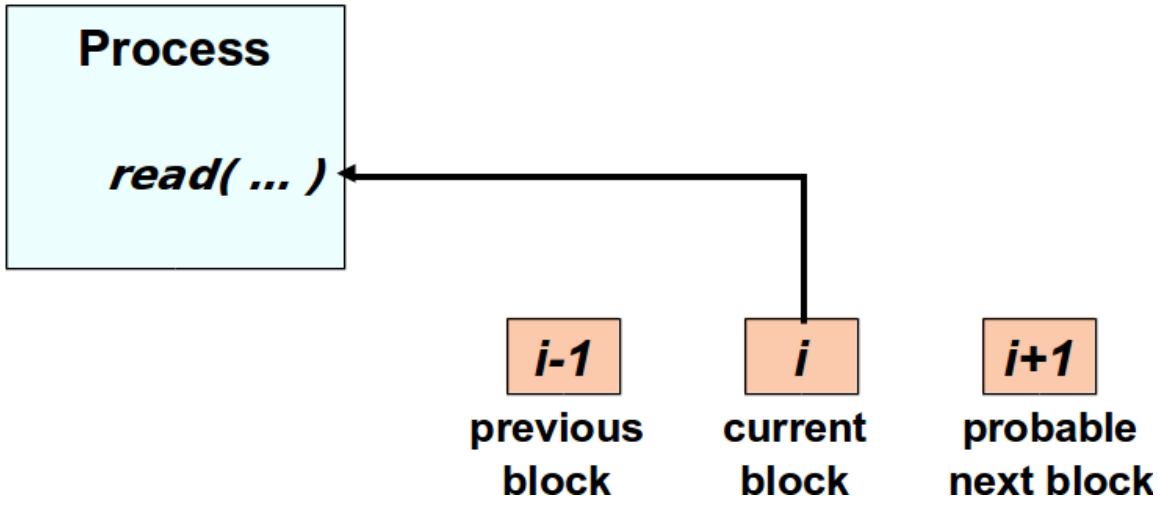


Buffer Cache

`Read(block)`

1. See if block present in buffer cache
   - If yes, then return buffer
2. Initiate disk read for the block
3. Sleep till read is complete
4. Return buffer

Read Ahead


Process
read(...)
i-1
previous block
i
current block
i+1
probable next block

Buffer Cache - Write

```
Write(block) // assume block in cache
```

- (Usually) Write to cache and return; the write to disk is done later (write-back cache)
- (Sometimes) Write to cache, schedule a write to disk and return (write-through cache)
- (Exceptional cases) Write to cache, do a synchronous (blocking) write to disk, and return

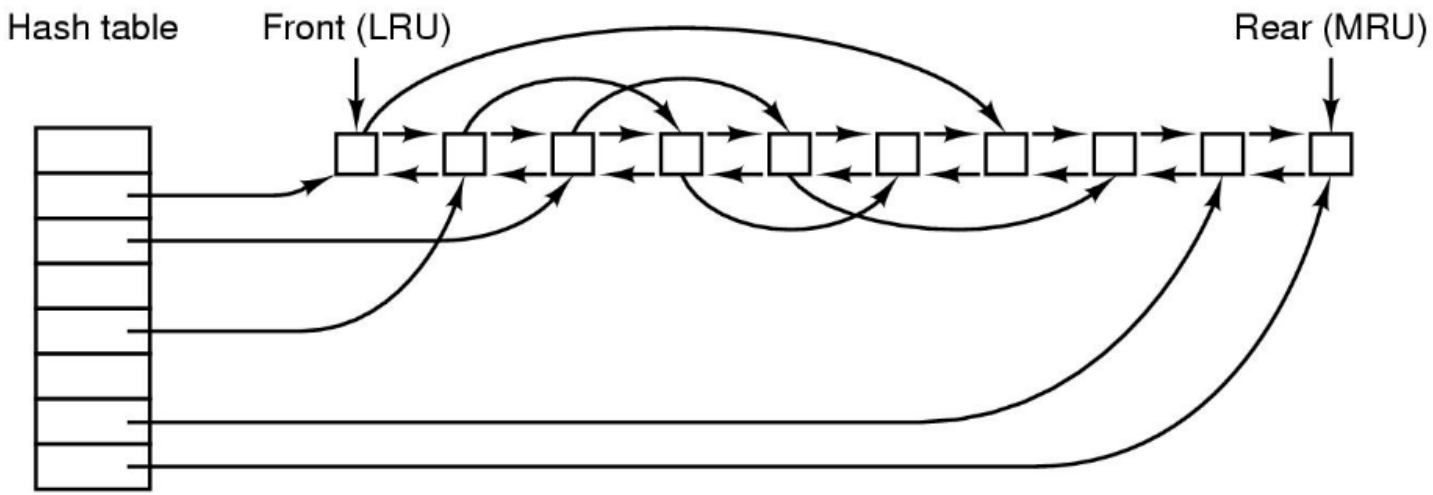
Write

- Write-back more efficient than write-through
- A disk crash might cause a more serious problem with write-back
- What happens when:
  - The system is turned off without a shutdown
  - A floppy is removed from the drive without unmounting
- System to the rescue: Every 30 seconds or do, a `sync` is done, writing all cache contents to disk

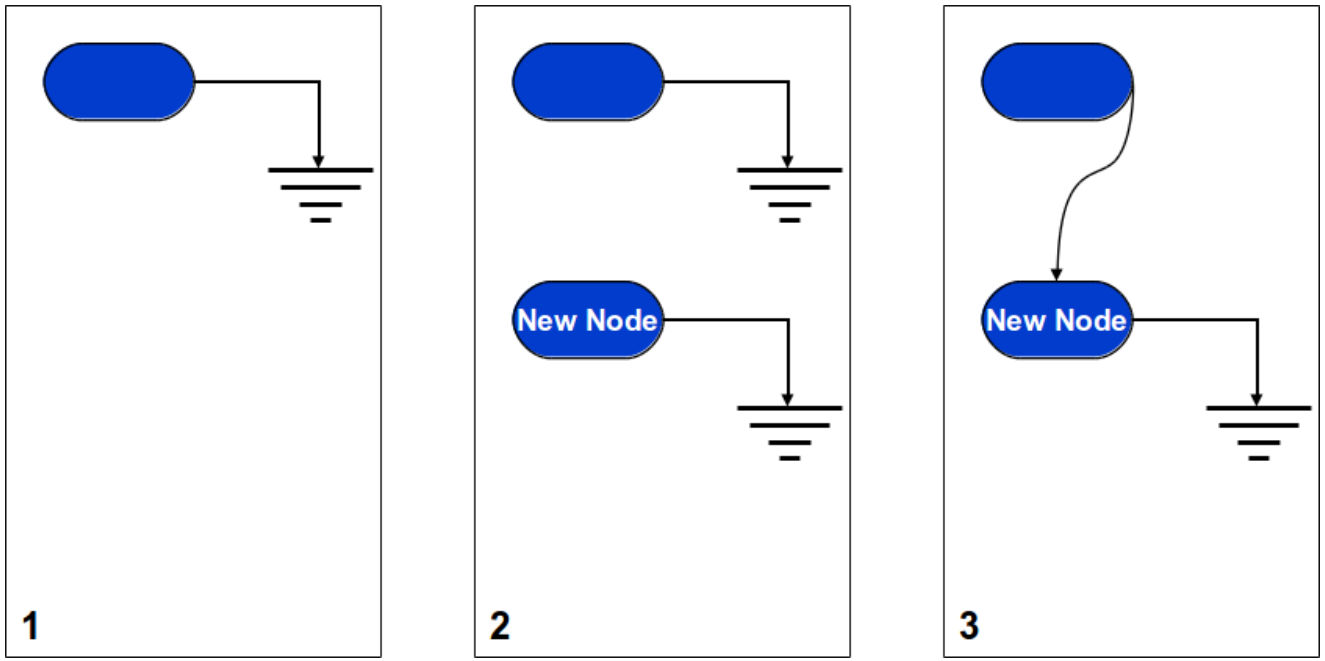
Structure of Cache

- Memory allocated by the system
- Lookup:
  - hash tables
- Page Replacement:
  - LRU
  - Keep a list sorted according to time of use

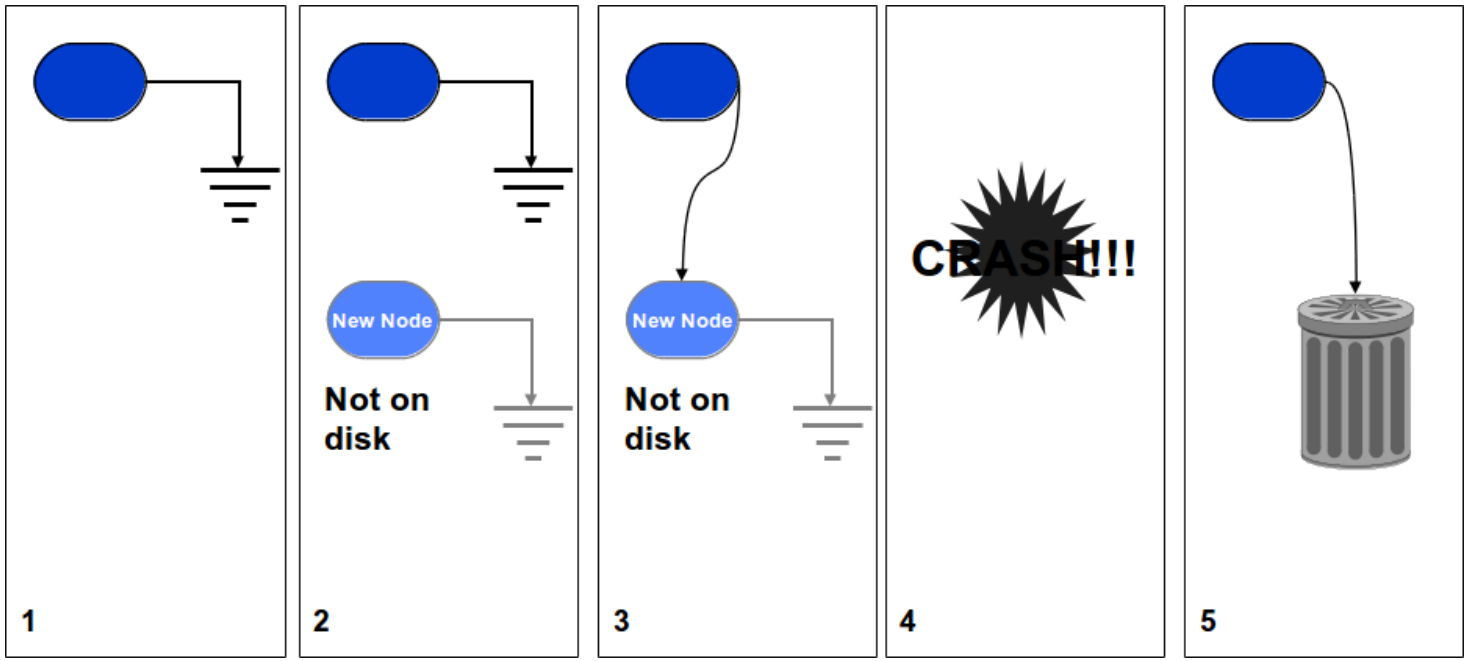
Structure of the Cache



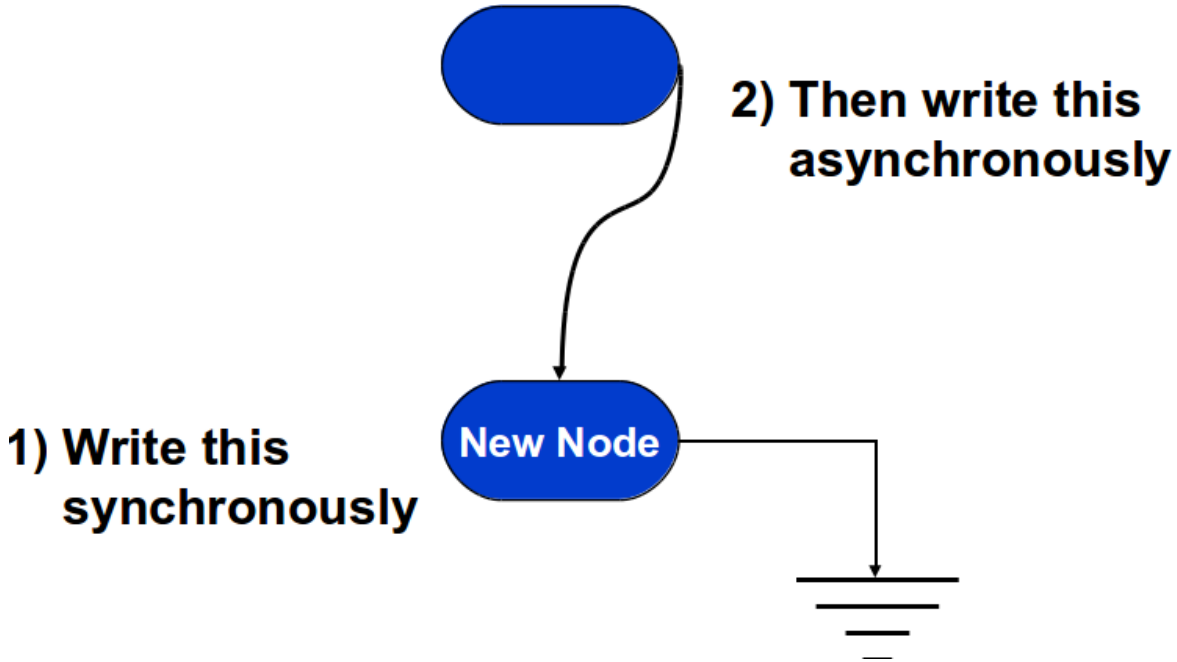
File-System Consistency (1)



File-System Consistency (2)



Keeping It Consistent



File Systems do Crash

- Bring to a consistent state using `fsck` on Unix
- Make sure every disk block in exactly one file or on the free list
- Go through all directories, and count the number of links per file - check for inconsistencies
- Might prompt the user before taking action

Log Structured File Systems

- If there's lots of caching, then most operations to the file system are *writes*
- Writes are quickest when there is no need to do a seek
- Thus: perform writes wherever the disk head happens to be