FILE SYSTEM API
File Names:

1. **inode number**
    1. Each file has exactly one unique inode number
    2. Different file systems may use the same number
    3. Numbers may be recycled after deletes
    4. Drawbacks:
        1. names hard to remember
        2. no organization or meaning to inode numbers
        3. semantics of offset across multiple processes
2. **paths**
    1. Directory Tree instead of single root directory
    2. File name (String names) needs to be unique only within a directory
    3. Still interact with inode numbers
    4. Store *path-to-inode* mappings in Directory
    5. **Drawbacks**: Expensive traversal
3. **file descriptor (FD)**
    1. Do expensive traversal once (open file)
    2. Store inode in descriptor object (kept in memory)
    3. Do reads/writes via descriptor, which tracks offset
    4. Advantages:
        1. human-readable names
        2. hierarchical
        3. traverse once
        4. offsets precisely defined

File functions:
open(): Create new files or open existing files
read()/write(): Access file contents
mkdir(): Create directories
lseek(): SEEK_SET, **set** offset ****to offset bytes; SEEK_CUR, set offset to cur location **plus** offset; SEEK_END, set offset to **size** plus offset
fork(): a parent creates a child and waits for it to complete. The child adjusts the current offset with lseek() and exits. The parent, after waiting for the child, checks the current offset and prints out its value. **unlink()**: path names are removed
close(): FDs are removed or process are quit
fsync(int fd): forces buffers to flush from memory to disk, tells disk to flush its write cache, Makes data **durable.** A successful close does not guarantee that the data has been successfully saved to disk, make sure use **fsync(2)**
rename(char *old, char *new): deletes an old link to a file, creates a new link to a file, Just changes name of file, does not move (or copy) data
Hard Links: another name for file

- increase ref count when link added
- does not remove file until ref count is 0

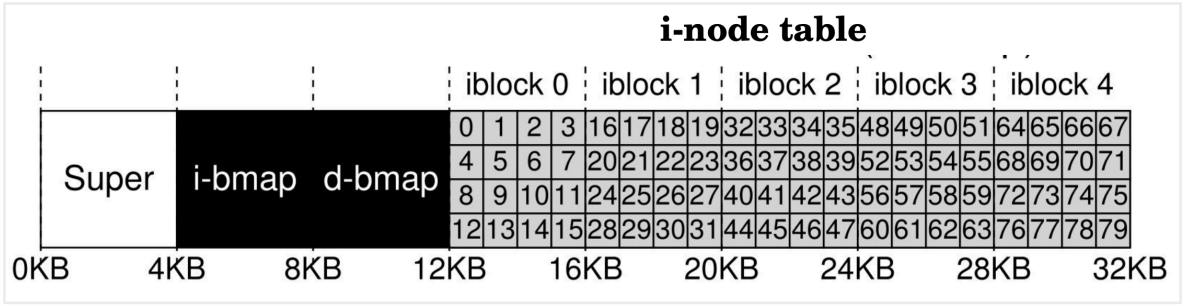
- cannot hard link directories

Soft Links: Point to second path name

- have new inode number
- Set bit in inode designating “soft link”;
- Interpret associated data as file name
- possible to link to directories
- does not change ref rount

File System

- Max # of files = max # of inodes = # of inode blocks * (sizeof(block)/ sizeof(inode));
- Inode table size = # of inode blocks * size of 1 block;
- block = (inum * sizeof(inode)) / block size;
- Sector # for fetching inode block = (block * block size + inodeStartAddr) or offset into inode region] / sector size = (inum * sizeof(inode) + inodeStartAddr) / sector size



Superblock: Parameters of the file system (e.g., how many inodes)
i-bitmap: Which inodes are in use?
d-bitmap: Which data blocks are in use?

FSCK and Journaling: Fix Inconsistencies

FSCK = file system checker

- **Strategy:** run checker after reboot/crash, scan entire file sys, find inconsistencies btwn bitmaps and inode, fix mismatches (Slow)

Journaling

- **Strategy**: Don’t delete any old info until all new info is safely on disk
  - Make a note of what needs to be written
  - After note is completely written, update file metadata and data
  - Remove note
  - If a note is not completely written, ignore note (old data still good)
  - If a note is completely written, *replay* it to recover data
- **Journal:** Blocks designated to store notes
- **Transaction:**
  - **TxB (“Begin Transaction”):** Holds unique id and blocks affected
  - **TxE (“End Transaction”):** Indicates transaction has committed
  - Set of writes that “belong together” (should execute atomically)
  - Last part of a transaction is its *commit block*
    - Transaction is considered *committed* after this block is written

- **Checkpoint:** Writing to in-place metadata and data after commit
- in real system:
  - Batch or group Transactions
  - For performance, many operations are placed in single transaction
  - Journal is large; treat as circular buffer
  - Checkpoint periodically
- **Optimizations**
- **Barriers**
  - Before journal commit, ensure journal transaction entries complete
  - Before checkpoint, ensure journal commit complete
  - Before free journal, ensure checkpoint (in-place updates) complete
- In last transaction block, store **checksum** of rest of transaction, During recovery: If checksum does not match, treat as not valid
- **Batched updates:** If two files are created, all things write twice, mark dirty in memory, update in one transaction
- **Delay checkpoints** some times to avoid system crush after journal write
- **Circular Log**
  - **Difficulty:** need to reuse journal space
  - **Solution:** keep many transactions for un-checkpointed data
- Avoid Journal Write Disk Twice
  - Journal all metadata, including superblock, bitmaps, inodes, indirects, directories
  - Guarantees metadata is consistent if crash occurs
  - Will not leak space or re-allocate blocks to multiple files
  - For regular user data, write it to in-place location whenever convenient
  - Files may contain **garbage** (partial old, partial new) if we crash and recover