Hon Wai Fung
CS162
4/8/09

Three set of table to manage file descriptor:

1. process open file table (one entry per file open)
    - File refer by index into table
    - Open file multiple time give multiple entry
    - 3 additional entry:
        - Standard In: 0
        - Standard Out: 1
        - Standard error 2
2. system open file table
    - points to inode for the file (index to the inode table)
3. Inode table (system wide)
    - Holding ative and recently used inodes

Sharing inode to share information and modification of other processes
System Open file table as an optimization (cache avoiding to go to directory each time opening file)
Kept in OS virtual memory area to avoid page out.

Directory: Map file name to file
Approach 1: Single directory for the whole disk. <name, index>

- Only one user can use a name
- long search
- people can see file name (security problem)

Approach 2: Separate directory for each user. <name, type, location(A,B,C disk letter)>

- limit name to 8 characters

Approach 3: Unix Approach: generalize directory structure to a tree

- directories are stored on disk as file (bit indicates as directory)
- <name, file descriptor index #>
    - Smith/course/CS162/Exams/EX1
- directory points to another directory
- special directory (root) (/ before directory name)
    - /home/Smith/course/CS162/Exams/EX1
- Descriptor #2 in Inode table points to root

directory compose of blocks of directory-block-size bytes

- (smaller than file and enough to be transferred to disk in one atomic operation)

file name is not the name of file, only refer to a file, a name the kernel search the file with.
Inode # is the name of the file (1 to 1 relationship)

Hard link: Pointer from directory to a file

- file are erase by removing a link to it. (only erase when the last link is erased)
- To really erase a file, have to put the file back into free block.
- Problem:
    - within file system, can't point to file in another disk (inode # local to file system)
    - Disk may not have been there

Symbolic link

- point to symbolic name for that file or directory
- When file does not exists, standard error

Working directory: directory you are currently in (temporary pointer, everything interpret w.r.t this directory)

Search path: file name that is not complete path name, this give the sequence of directory to find

If link count goings to zero, just garabage collect and put back onto free list

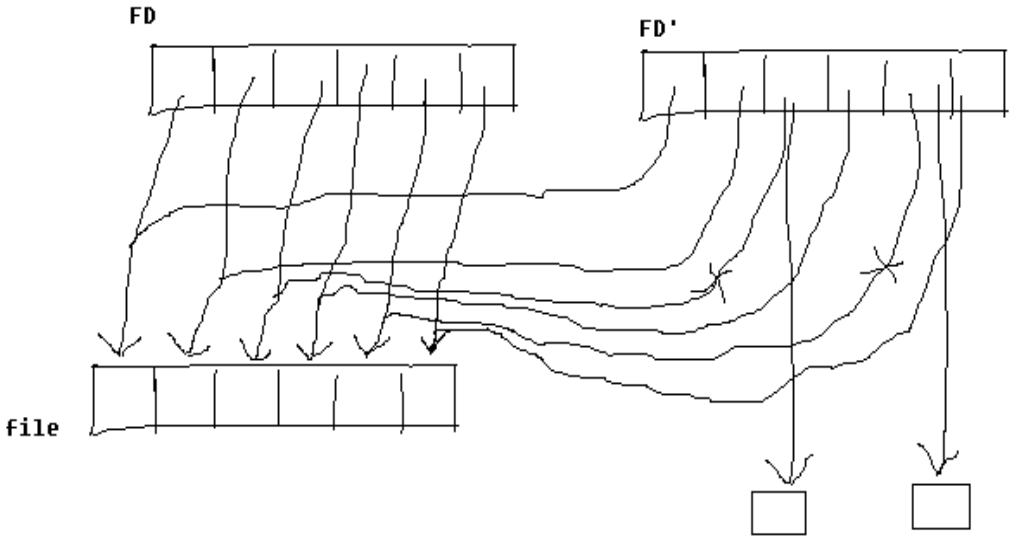
Commands:
- Read: read a record from the file (OS read blocks of data, but only give next record)
- Write: require disk space allocate
- Rename: rename file (Unix may/maynot move file from one directory to another)
  - If new name already exist, destroy the old one
- Seek: move to a location in file
- Synch: move blocks from disk cache to disk
- Change property
- link
- lock & unlock
- truncate

Pseudo File (Treat I/O devices as files)

File Backup and Recovery
- File Lost:
  - System Crash
    - Physical Hard failture
    - Softwarefailure
    - General System failure while file is open (usually power failure)
  - User Error
    - Want to get file back after we destroyed them
  - Sabotage and malicious users

Approaches:
- Periodic full dump: periodically dump all file to a backup storage
  - (Checkpoint dump)
  - Large amount of data, slow to dump, large space
- Incremental (periodic) dump: dump only modified file (when file close or user logs out)
  - Only lose file when file is open
  - Large amount of data
  - long recovery process
  - (May dump whole file, or dump only block that is modified)

If system crash while using, file system maybe left in inconsistent state.
- Solution:
  - Work on copy of a file and swap it for the original when close.
    - Multiple people write on a file, only last one that save will remain
    - need write back to atomic (avoid failling while writing back)
  - Write a log of all changes to file (can back up)(audit trail)
  - Write a list of changes to file prior to modifying the file
  - Keep multiple copies of the file (careful replacement)
  - Make a new copy of any part of the file as it is modified,
    - replace old parts with new parts when we close file.
    1. Duplicate file descriptor.
    2. Update new copy of file descriptor.
    3. When write, point to new block.
    4. Swap file descriptor when closed.


FD
FD'
file