CS323 Operating Systems
File System II

Yuanyuan Zhou

Lecture 24

3/21/2003

Content of this lecture

- Administrative announcements
- Disk scheduling
- File systems basic concepts
- Summary
- The hardest question:
  - Using swap to implement mutex

Administatives

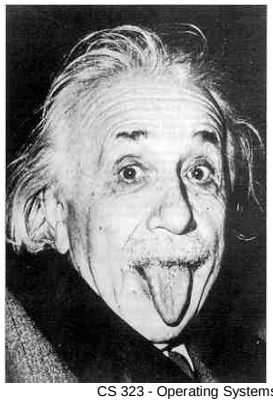
- Regrading Period
  - MP1: until this Friday, 3/21 5pm
  - Midterm1: until Friday, 3/21 5pm
    - Pick/up your midterm from TA's office
  - Submit written request to TA
  - After this deadline, no regrading request will be granted!!
- Quiz3 closes today.
- 3/31 & 4/2 lectures
  - Given by TA: Jeff

Review

- Disk scheduling
  - FCFS
  - Shortest seek time first
  - Elevator (SCAN)
  - C-SCAN
- File system basic concepts
  - File
  - Attributes
  - Types

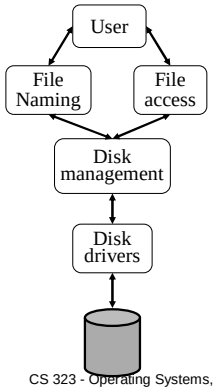
So What Makes Filesystems Hard?

- Files grow and shrink in pieces
- Little *a priori* knowledge
- 6 orders of magnitude in file sizes
- Overcoming disk performance behavior
- Desire for efficiency
- Coping with failure



File System Components

- Disk management
  - Arrange collection of disk blocks into files
- Naming
  - User gives file name, not track or sector number, to locate data
- Security
  - Keep information secure
- Reliability/durability
  - When system crashes, lose stuff in memory, but want files to be durable



Some Definitions

- File descriptor (fd)
  - an integer used to represent a file – easier than using names
- Metadata
  - data about data - bookkeeping data used to eventually access the “real” data
- Open file table
  - system-wide list of descriptors in use

Kinds of Metadata

- inode – index node, or a specific set of information kept about each file
  - Two forms – on disk and in memory
- Directory – names and location information for files and subdirectories
  - Note: stored in files in Unix
- Superblock – contains information to describe the file system, disk layout
- Information about free blocks/inodes on disk

Contents of an Inode

- Disk inode:
  - File type, size, blocks on disk
  - Owner, group, permissions (r/w/x)
  - Reference count
  - Times: creation, last access, last mod
  - Inode generation number
  - Padding & other stuff
- 128 bytes on classic Unix

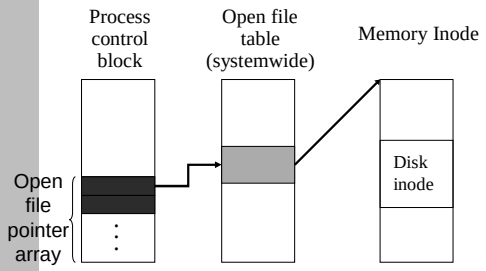
Directories in Unix

- Stored like regular files
  - Contents are file names and inode #s
  - Names are nul-terminated strings
- Logic
  - Separates file from location in tree
  - File can appear in multiple places
- What are the drawbacks?

Effects of Corruption

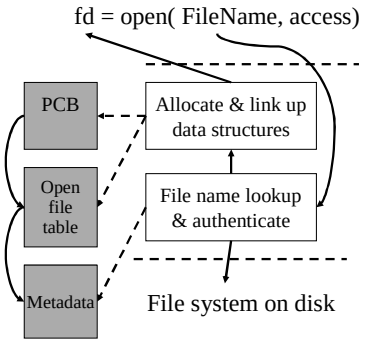
- inode – file gets “damaged”
  - Maybe some “free” block gets viewed
- Directory – “lose” files/directories
  - Might get to read deleted files
- Superblock – can’t figure out anything
  - This is why we replicate the superblock

Data Structures for A Typical File System



Opening A File

- File name lookup and authentication
- Copy the file metadata into the in-memory data structure, if it is not in yet
- Create an entry in the open file table (system wide) if there isn't one
- Create an entry in PCB
- Link up the data structures
- Return a pointer to user



Open-file Table Information

- File Pointer
  - current file position pointer
- File Open Count
  - counter which tracks the number of opens and closes. If files are closed, the OS must reuse the space in open-file table. Because multiple processes may open the same file, OS must wait until the last file closes before removing the entry.
- Disk Location
  - information needed to locate the file on disk.

Reading And Writing

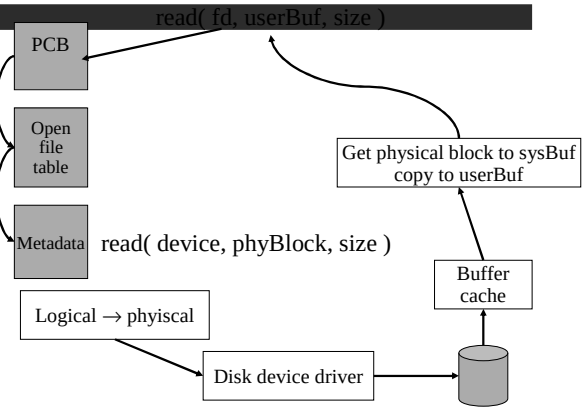
What happens when you…

- read 10 bytes from a file?
- write 10 bytes into an existing file?
- write 4096 bytes into a file?

Disk works on blocks (sectors)

- Can have temporary (ephemeral) buffers
- Longer lasting buffers = disk cache

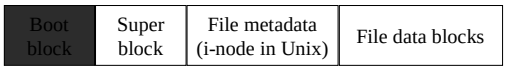
Reading A Block



Other File Issues

- Memory Mapping a File
  - Associate a part of the VM space with a section of a file.
  - Reads and writes to that memory region are then treated as reads and writes to the file.
- Internal File Structure
  - mapping between logical record and physical record
  - packing a number of logical records into physical blocks.
  - The logical record size, physical block size and packing technique determine how many logical records are in each physical block.
- Consistency Semantics
  - important criterion for evaluation of any file system that supports sharing of files.

A Disk Layout for A File System



- Superblock defines a file system
  - size of the file system
  - size of the file descriptor area
  - free list pointer, or pointer to bitmap
  - location of the file descriptor of the root directory
  - other meta-data such as permission and various times
- For reliability, replicate the superblock

File Usage Patterns

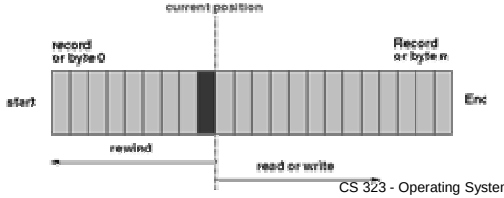
- How do users access files?
  - Sequential: bytes read in order
  - Random: read/write element out of middle of arrays
  - Whole file or partial file
- How are files used?
  - Most files are small
  - Large files use up most of the disk space
  - Large files account for most of the bytes transferred
- Bad news
  - Need everything to be efficient

Data Structures for Disk Management

- A “header” for each file (part of the file meta-data)
  - Disk sectors associated with each file
- A data structure to represent free space on disk
  - Bit map
    - 1 bit per block (sector)
    - blocks numbered in cylinder-major order, why?
  - Linked list
  - Others?
- How much space does a bit map need for a 4G disk?

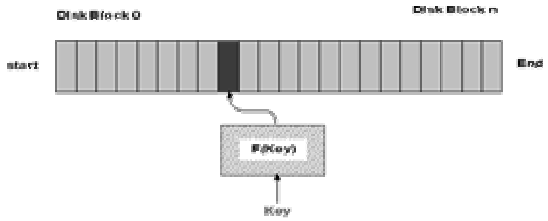
Sequential File Organization: Tape

- Organized like a tape; can be on a tape or direct access device.
- The records are stored and also retrieved in physical order.
- The ``next'' record physically follows the current one.
- Inserting a record means rewriting the file



Directed File Access Organization: Disk

- User maps key into disk address
- Access in physical order or random
- No notion of next record

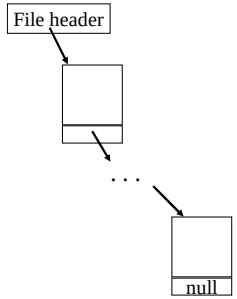


File Allocation

- Contiguous
- Non-contiguous (linked)
- Tradeoffs?

Linked Files

- File header points to 1st block on disk
- Each block points to next
- Pros
  - Can grow files dynamically
  - Free list is similar to a file
- Cons
  - random access: horrible
  - unreliable: losing a block means losing the rest



Contiguous Allocation

- Request in advance for the size of the file
- Search bit map or linked list to locate a space
- File header
  - first sector in file
  - number of sectors
- Pros
  - Fast sequential access
  - Easy random access
- Cons
  - External fragmentation
  - Hard to grow files

Reminder