CIS3110 Lecture 9 - Summary Notes: Disk Structure and File Systems

1. Disk Structure and Addressing (Chapter 11.1, 11.2)

Three-Dimensional Disk Addressing

- Disks use a three-dimensional addressing scheme:
  - **Theta**: Angular position (sectors around the disk)
  - **R**: Radius (tracks from inner to outer edge)
  - **Z**: Head position (which platter surface)
- This forms a "cubical" addressing space despite the circular physical shape
- Standard sector size: 512 bytes (hardware-defined unit)

Physical Disk Components

- Hard drives contain multiple platters with read/write heads
- Heads are positioned by movable arms over the spinning platters
- Modern drives: multiple platters in sealed cases
- Historical perspective: early drives used removable "cartridges" with platters

2. Disk Scheduling Algorithms (Chapter 12.4)

Problem: Multiple Disk Requests

- Moving the disk arm is costly in terms of time
- Goal: Minimize head movement while serving all requests

Algorithms

1. **FIFO (First-In, First-Out)**
   - Serve requests in order of arrival
   - Problem: Inefficient movement, excessive head travel
2. **Shortest Job First (Shortest Seek Time First)**
   - Always go to the closest request
   - Problem: Starvation for distant requests
3. **Elevator/SCAN Algorithm**
   - Similar to modern elevator operation

- Move arm in one direction until no more requests in that direction
- Then reverse direction and repeat
- Picks up any new requests that are "ahead" of current position
- Guarantees service to all requests eventually
- Better utilization of head movement

3. Block Size Considerations (Chapter 11.2, 11.3)

Block vs. Sector

- **Sector**: Hardware-defined unit (typically 512 bytes)
- **Block**: File system unit (comprised of multiple sectors)

Block Size Trade-offs

- **Larger blocks**:
  - More efficient transfers (more data per seek)
  - Worse internal fragmentation (wasted space for small files)
- **Smaller blocks**:
  - Less wasted space
  - More seeking and indexing overhead

4. FAT File System (Chapter 14.3, 14.4)

Structure

1. **Master Boot Record (MBR)**
   - First block on disk
   - Contains:
     - Jump location for boot code
     - Sector size information
     - Number of FAT copies
     - Disk geometry (sectors per track, heads, etc.)
     - Root directory entries count
     - Media type code
     - OEM name
2. **File Allocation Table (FAT)**

    - Two identical copies for redundancy (protection against corruption)
    - Acts as a linked list of block pointers
    - Each entry corresponds to one logical block on disk
    - Special values for "end of file" and "free block"
3. **Root Directory**
    - Fixed-size array of directory entries (limited to 1024 entries)
    - Each entry contains:
        - 8.3 format filename (8 chars + 3 char extension)
        - File attributes (read-only, hidden, etc.)
        - Modification time
        - File size in bytes
        - First block number
4. **Data Blocks**
    - Rest of the disk stores actual file data
    - Subdirectories are just special files with directory entries

FAT as a File Structure

- Files are represented as linked lists of blocks
- FAT table entries serve as "next pointers"
- Parallel structure: data blocks contain content, FAT contains linking information
- Disadvantages:
    - Poor random access (must traverse list from beginning)
    - Disk fragmentation issues

Historical Evolution of Block Sizes

- As disk sizes increased, Microsoft increased cluster (block) sizes
- For 16-bit FAT:
    - 360K disk: 1-2 sectors per cluster
    - 4GB disk: 64KB clusters
    - 16GB disk: 256KB clusters
- Larger cluster sizes lead to significant space wastage for small files
- NTFS follows similar pattern but supports much larger disks (up to 256TB)

5. Unix File System Overview (Chapter 15.5, 15.6)

Inode-Based Structure

- Uses a tree structure instead of linked list
- Root of file structure is the **inode** (index node)

Components

1. **Super Block**
   - Contains file system parameters
   - Multiple redundant copies throughout disk
   - Includes:
     - First data block offset
     - File system size
     - Root inode location
     - Free space information
     - Device geometry
     - Dirty flag for caching
2. **Inodes**
   - Complete descriptor for each file
   - Contains:
     - File mode (permissions: read, write, execute)
     - Link count (number of directory entries pointing to this inode)
     - Owner/Group IDs
     - File size
     - Access/modification times
     - Direct block pointers (first 6 in example)
     - Indirect block pointers (for larger files)
3. **File Structure (Tree-based)**
   - Small files: Direct block pointers in inode
   - Medium files: First indirect pointer (addresses ~1024 more blocks)
   - Large files: Second indirect pointer (addresses ~1024^2 more blocks)
   - Very large files: Third indirect pointer (addresses ~1024^3 more blocks)

Advantages over FAT

- Efficient random access (can directly access any block)
- Short, broad tree structure minimizes seek time
- Multiple hard links possible (same file appears in multiple directories)

6. SSD vs. Traditional Hard Drives

- SSDs don't have mechanical heads, so seek time isn't a factor
- Addressing structure remains similar, but physical layout differs
- Elevator/SCAN algorithms less relevant for SSDs

Next Lecture

- More details on Unix file system
- Comparison with NTFS
- Additional features of inodes (link count usage)