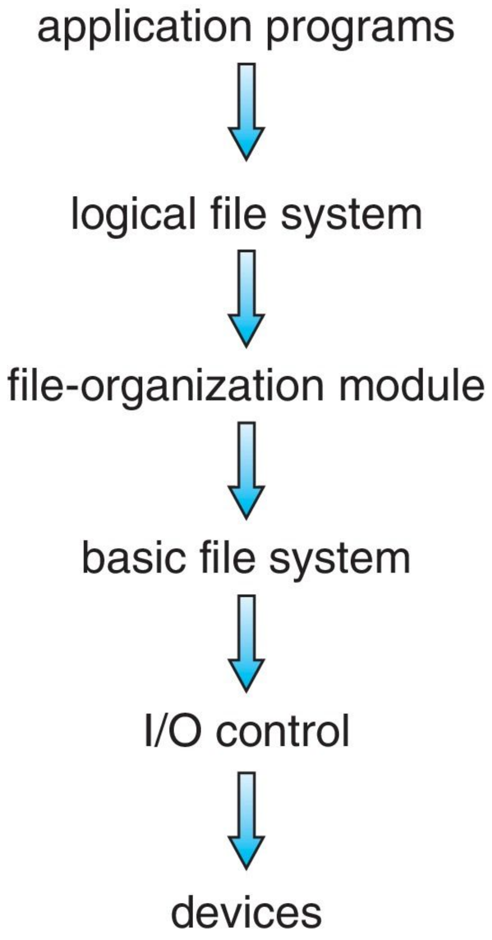
16. File System Implementation

CS 4352 Operating Systems

Adopted from slides of Drs. Silberschatz, Huang, & Bryant

Layered File System

- File system is often organized into layers



File System Layers

- Device drivers manage I/O devices at the I/O control layer
    - Given commands like “read drive1, cylinder 72, track 2, sector 10, into memory location 1060” outputs low-level hardware specific commands to hardware controller
- Basic file system given command like “retrieve block 123” translates to device driver
    - Also manages memory buffers and caches (allocation, freeing, replacement)
    - Buffers hold data in transit
    - Caches hold frequently used data
- File organization module understands files, logical address, and physical blocks
    - Translates logical block # to physical block #
    - Manages free space, disk allocation

File System Layers (Cont.)

- Logical file system manages metadata information
  - Translates file name into file number, file handle, location by maintaining file control blocks (inodes in UNIX)
    - File control block (FCB) – a structure containing detailed information about a file
  - Directory management
  - Protection
- Layering is useful for reducing complexity and redundancy, but adds overhead and can decrease performance

On-Disk File System Structures

- Boot control block contains info needed by system to boot OS from that volume
  - Needed if volume contains OS, usually first block of volume
- Volume control block (superblock, master file table) contains volume details
  - Total # of blocks, # of free blocks, block size, free block pointers or array
- Directory structure organizes the files
  - Names and inode numbers, master file table
- Per-file File Control Block (FCB) contains many details about the file
  - They are inodes in UNIX

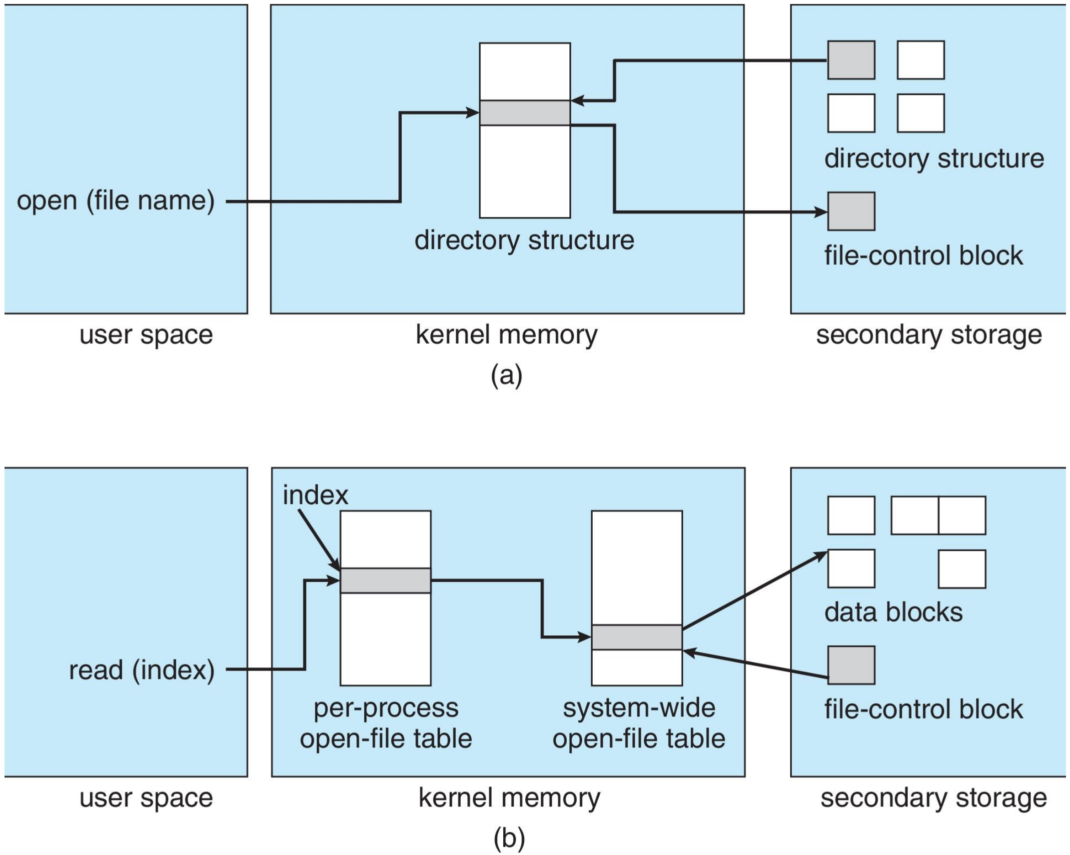
FCB

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

In-Memory File System Structures

- **Mount table** stores file system mounts, mount points, file system types
- **System-wide open-file table** contains a copy of the FCB of each file and other info
- **Per-process open-file table** contains pointers to appropriate entries in system-wide open-file table as well as other info
- Buffers and caches hold data blocks from secondary storage

Open and Read a File



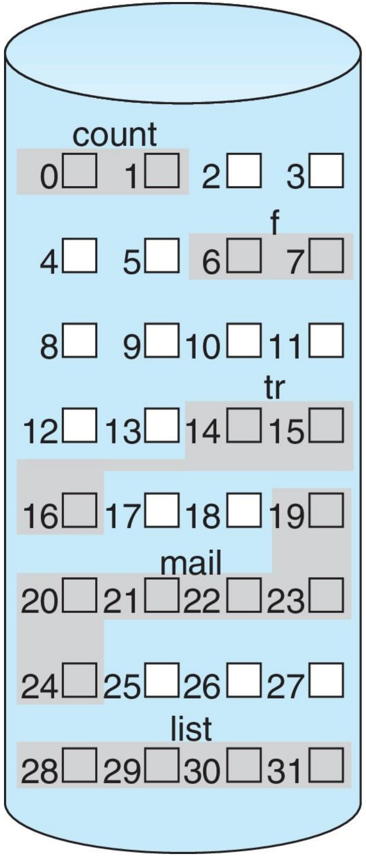
Directory Implementation

- Linear list of file names with pointer to the data blocks
  - Simple to program
  - Time-consuming to execute
    - Linear search time
  - Could keep ordered alphabetically via linked list or use B+ tree
- Hash Table – linear list with hash data structure
  - Decreases directory search time
  - Collisions – situations where two file names hash to the same location
  - Only good if entries are fixed size, or use chained-overflow method

Allocation Methods - Contiguous

- An allocation method refers to how disk blocks are allocated for files
- Contiguous allocation – each file occupies set of contiguous blocks
  - Best performance in most cases
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Problems include finding space for file, knowing file size, external fragmentation, need for **compaction** off-line (downtime) or on-line

Contiguous Allocation



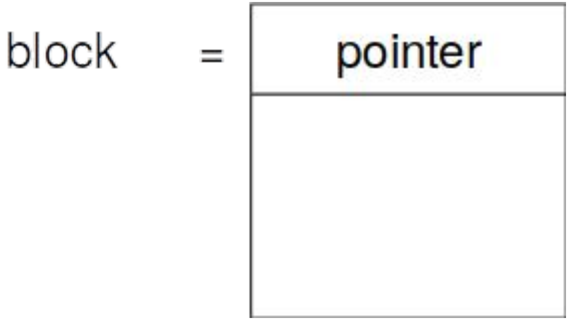
directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Extent-Based Systems

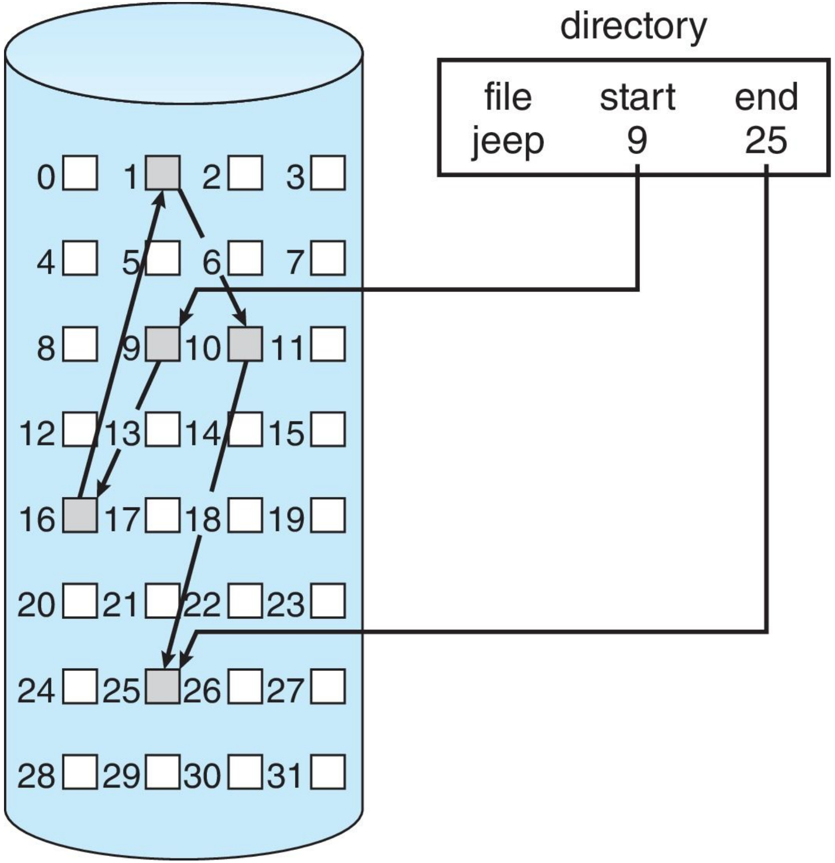
- Many newer file systems (i.e., Veritas File System) use a modified contiguous allocation scheme
- Extent-based file systems allocate disk blocks in extents
- An extent is a contiguous block of disks
    - Extents are allocated for file allocation
    - A file consists of one or more extents

Allocation Methods - Linked



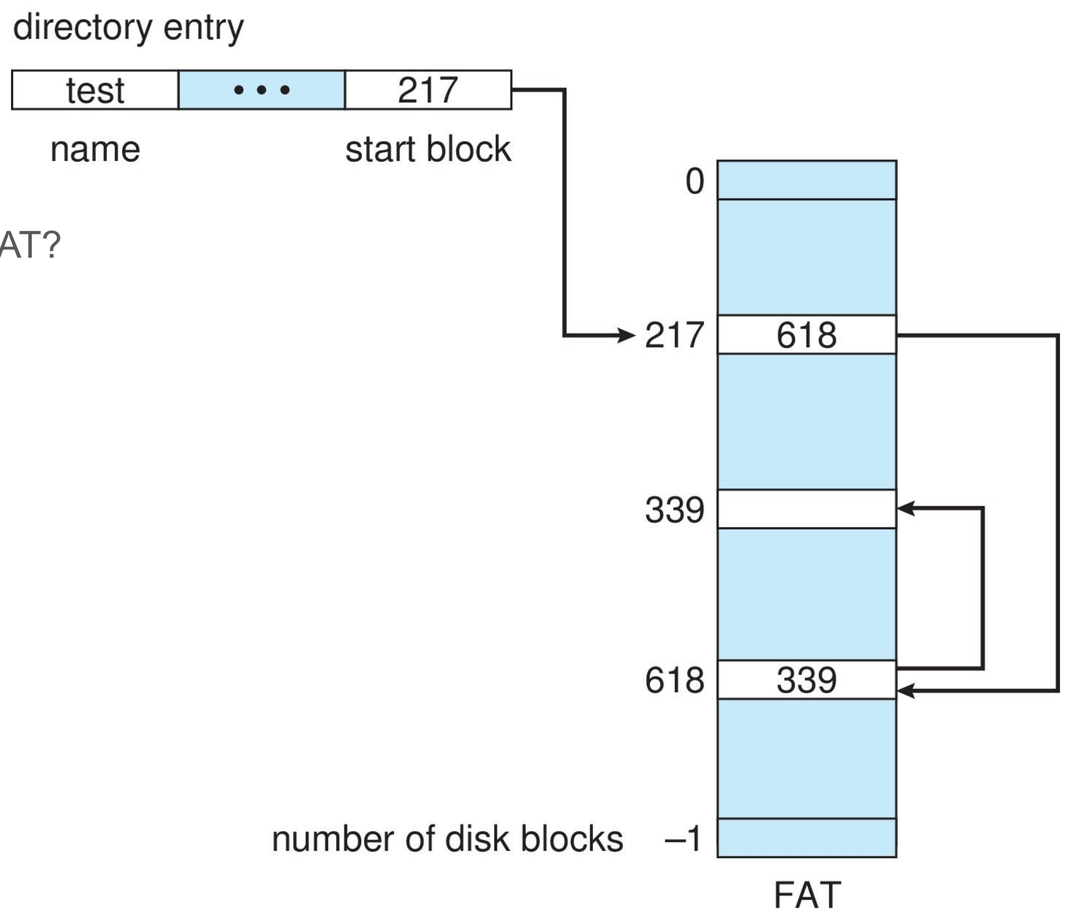
- Linked allocation – each file a linked list of blocks
  - File ends at null pointer
  - No external fragmentation
  - Each block contains pointer to next block
  - No compaction
  - Free space management system called when new block needed
  - Improve efficiency by clustering blocks into groups but increases internal fragmentation
  - Reliability can be a problem
    - E.g., one block in the middle is corrupted
  - Locating a block can take many I/Os and disk seeks
- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk

Linked Allocation



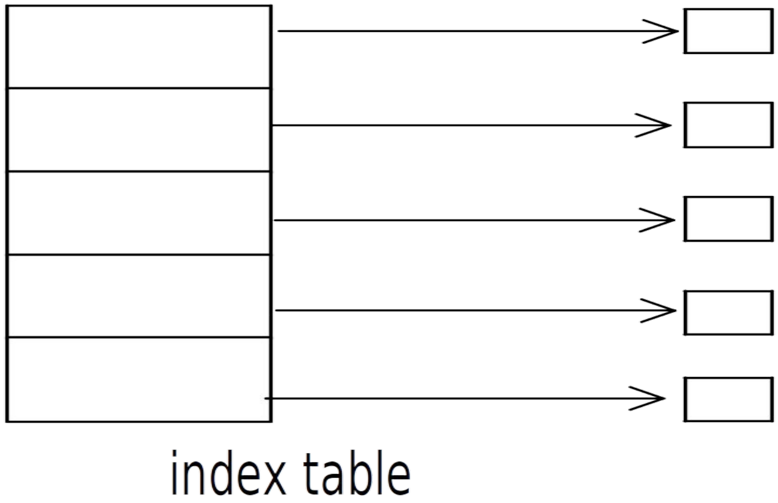
File-Allocation Table

- Entry size = 16 bits
  - What's the maximum size of the FAT?
    - 65,536 entries
  - Given 512B blocks, what's the maximum size of FS?
    - 32MB
- FAT32 uses 32 bit entry

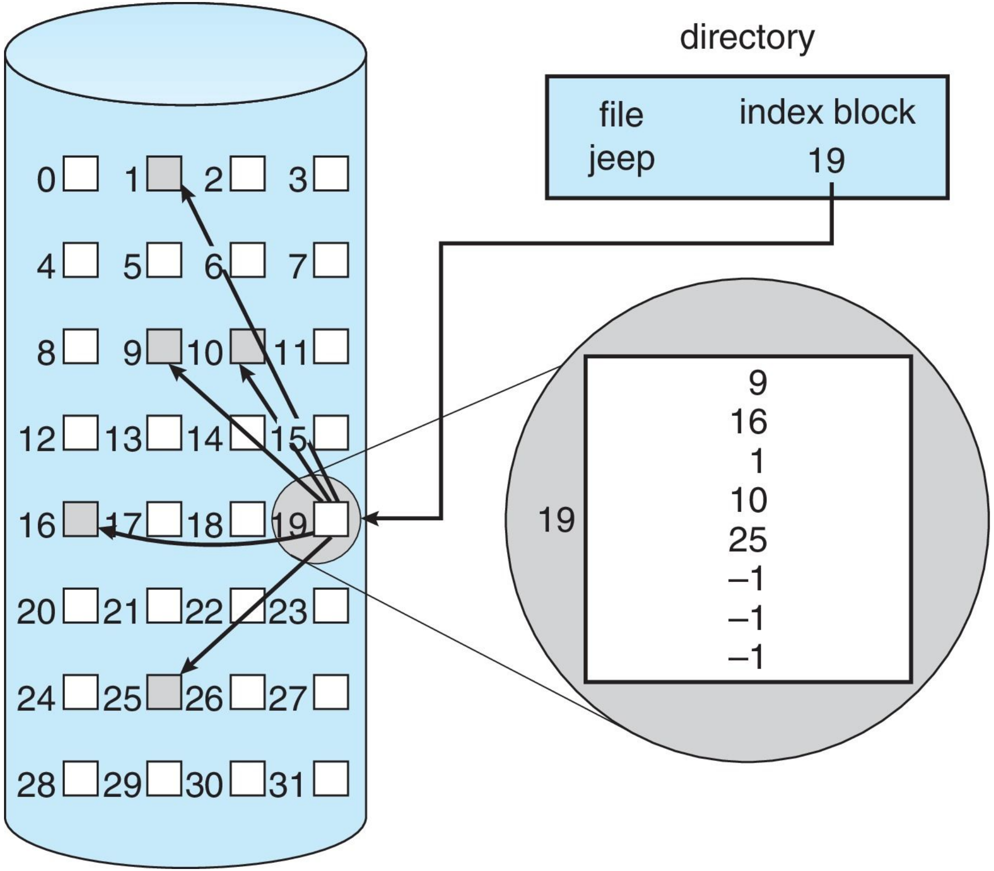


Allocation Methods - Indexed

- Indexed allocation
  - Each file has its own index block(s) of pointers to its data blocks
- Logical view



Example of Indexed Allocation



Unix/Linux inodes

- Index nodes (inodes) contains the following metadata about a file
  - File type (for example, regular, char device, block device, directory, symbolic link)
  - Owner of the file
  - Group of the file
  - File access permissions (user, group, other)
  - Three timestamps:
    - Time of last access (ls -lu)
    - Time of last modification (default timestamp in ls -l)
    - Time of last status change (change to inode info) (ls -lc)
  - Number of hard links to file
  - Size of the file (in bytes)
  - Number of blocks allocated to file
  - Pointers to the data blocks

Linux ext2

- 15 pointers
  - Pointers 0 to 11 point to direct blocks
  - Pointer 12 points to an indirect block
  - Pointer 13 points to a doubly indirect block
  - Pointer 14 points to a triply indirect block
  - Each pointer is 32-bit
- Small files fit entirely in direct pointers
- Bigger files use:
  - Indirect pointers
  - Double-indirect pointers
  - Triple-indirect pointers

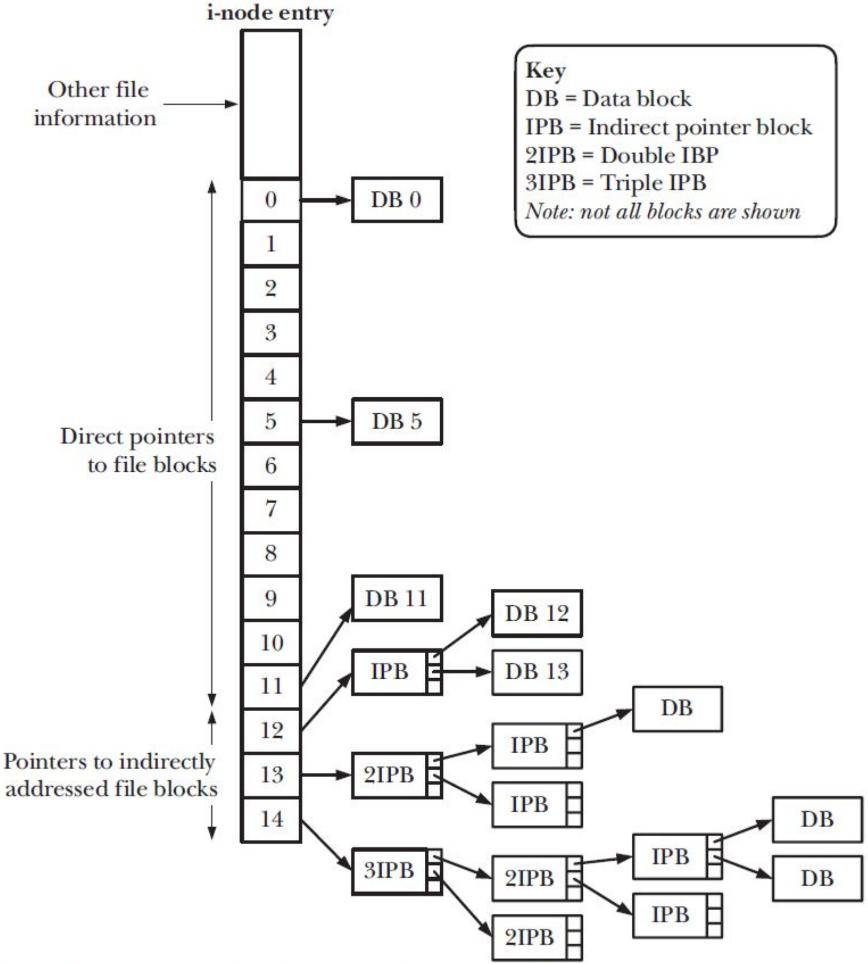


Figure 14-2: Structure of file blocks for a file in an *ext2* file system

Unix/Linux Directories

- A directory is stored in a filesystem in a similar way as a regular file, but it is marked as a directory in its inode
- It's a file with a special organization
  - It's a table consisting of filenames and inode numbers
- Where do you start looking?
  - Root directory always inode #2
  - (0 and 1 historically reserved)

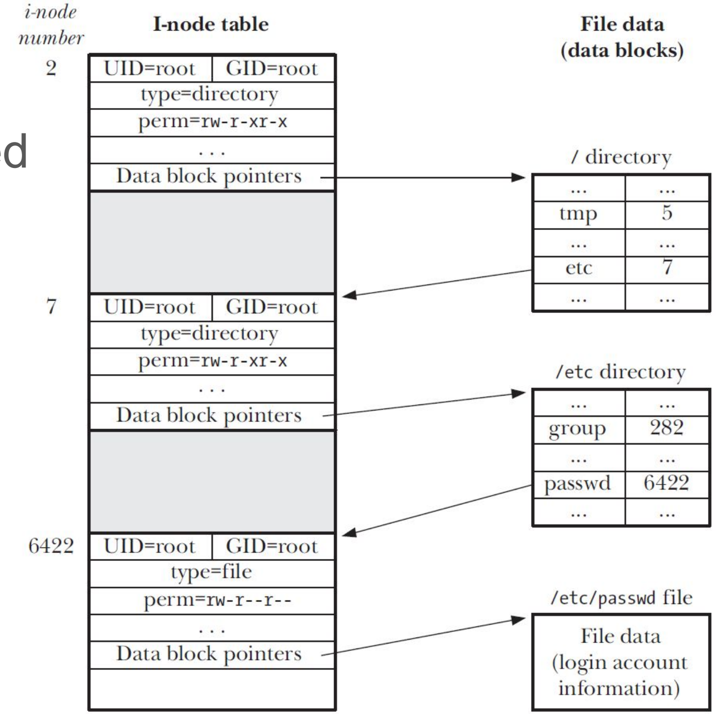


Figure 18-1: Relationship between i-node and directory structures for the file /etc/passwd

Performance

- Best method depends on file access type
  - Contiguous great for sequential and random
- Linked good for sequential, not random
  - Declare access type at creation -> select either contiguous or linked
- Indexed more complex
  - Single block access could require some index block reads then data block read
  - Clustering can help improve throughput, reduce CPU overhead
- For NVM, no disk head so different algorithms and optimizations needed
  - Using old algorithm uses many CPU cycles trying to avoid non-existent head movement
  - With NVM goal is to reduce CPU cycles and overall path needed for I/O

Free-Space Management

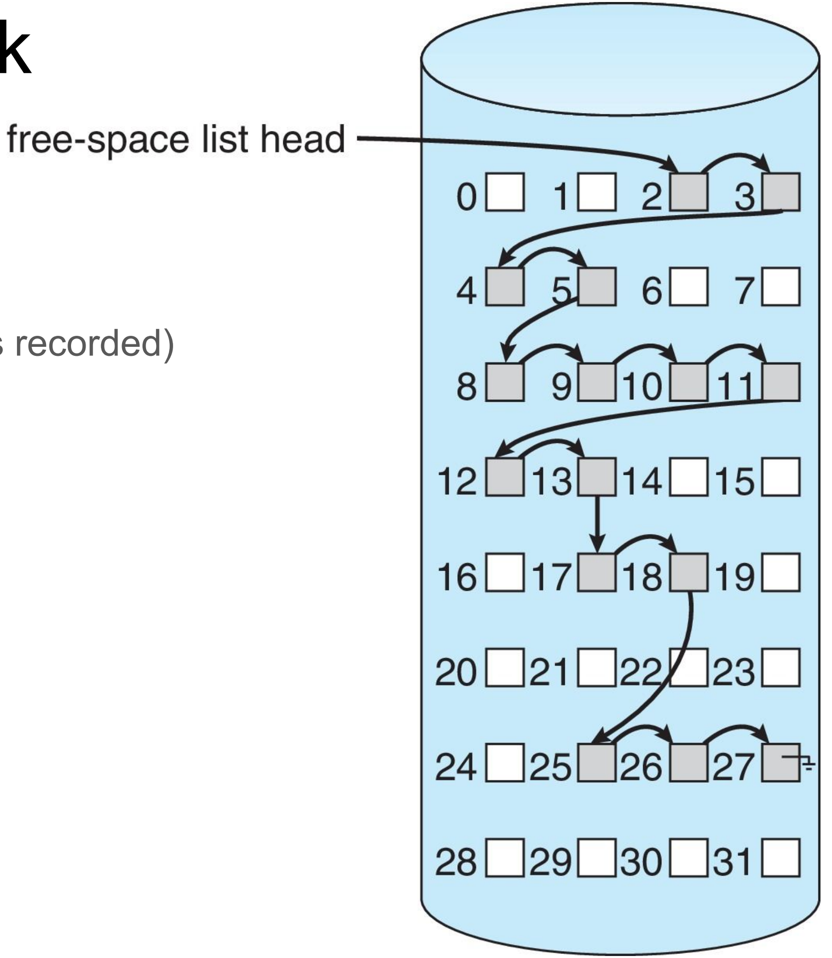
- File system maintains free-space list to track available blocks/clusters
    - (Using term “block” for simplicity)
- The free-space list is often implemented as a bitmap
    - If the block is free, the corresponding bit is 1
    - If the block is allocated, the corresponding bit is 0
- The free-space list can be implemented as a linked list
    - The file system keeps a pointer to the first free block
        - Free-space list head
    - FAT uses this method incorporated into the table

Bitmap Method

- Block number calculation:
  - (number of bits per word) * (number of 0-value words) + offset of first 1 bit
- Bitmap requires extra space
  - Example:
    - block size = 4KB = 2^12 bytes
    - disk size = 2^40 bytes (1TB)
    - n = 2^40/2^12 = 2^28 bits (or 32MB)
    - If clusters of 4 blocks -> 8MB of memory
- Easy to get contiguous files

Linked Free Space List on Disk

- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
  - No need to traverse the entire list (if # free blocks recorded)



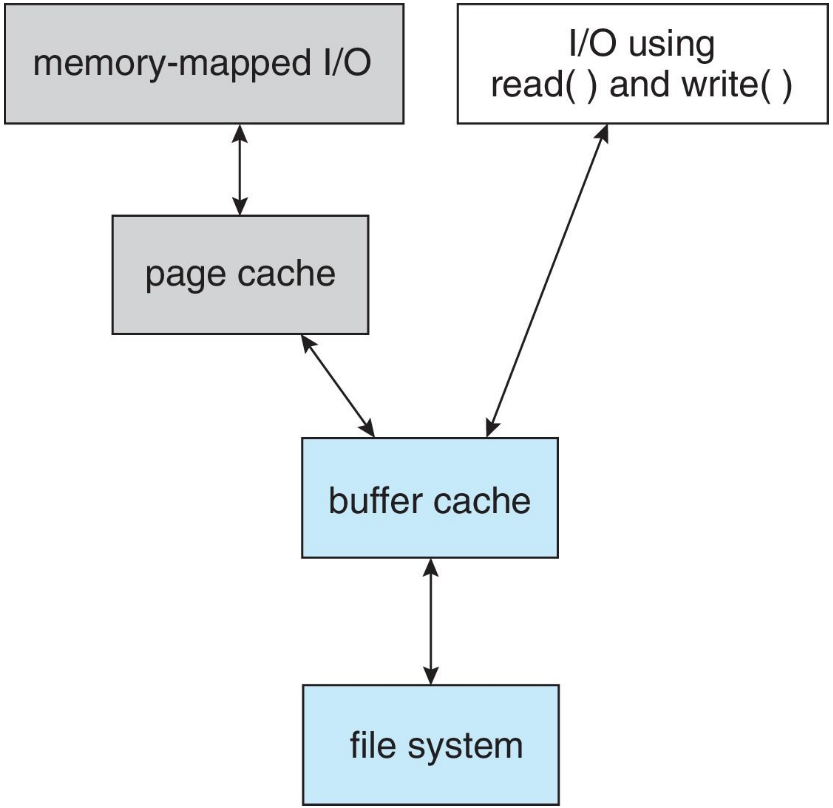
Other Methods

- Grouping
  - Modify linked list to store the address of next n-1 free blocks in first free block, plus a pointer to next block that contains free-block-pointers
- Counting
  - Because space is frequently contiguously used and freed, with contiguous-allocation allocation, extents, or clustering
    - Keep address of first free block and count of following free blocks
    - Free space list then has entries containing addresses and counts

Page Cache

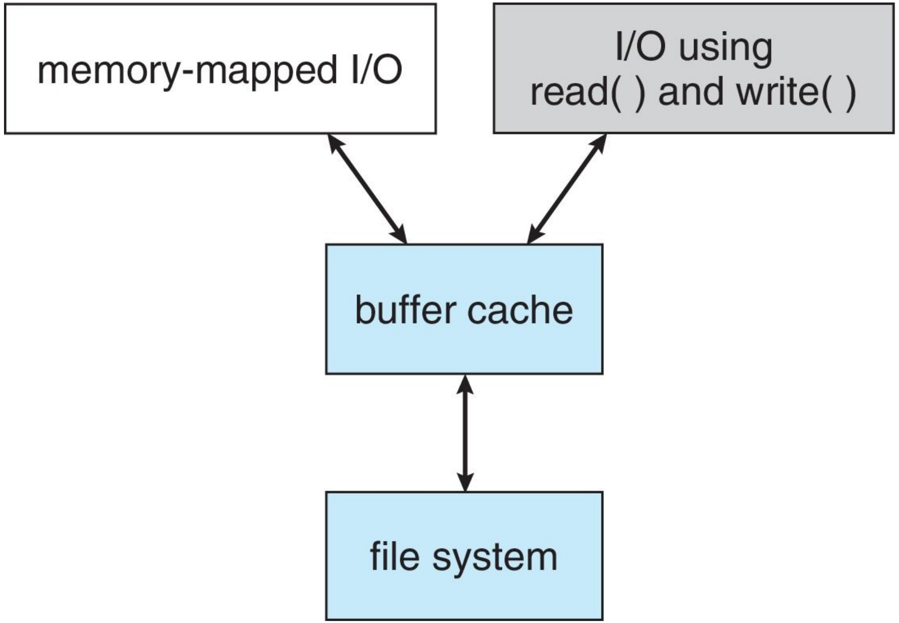
- A **page cache** caches pages rather than disk blocks using virtual memory techniques and addresses
- Memory-mapped I/O uses a page cache
- Routine I/O through the file system uses the buffer (disk) cache

I/O Without a Unified Buffer Cache



Unified Buffer Cache

- A **unified buffer cache** uses the same page cache to cache both memory-mapped pages and ordinary file system I/O to avoid **double caching**



Recovery

- Consistency checking – compares data in directory structure with data blocks on disk, and tries to fix inconsistencies
  - Can be slow and sometimes fails
- Use system programs to back up data from disk to another storage device (magnetic tape, other magnetic disk, optical)
- Recover lost file or disk by restoring data from backup

Log Structured File Systems

- Log structured (or journaling) file systems record each metadata update to the file system as a **transaction**
- All transactions are written to a log
  - A transaction is considered committed once it is written to the log (sequentially)
  - However, the file system may not yet be updated
- The transactions in the log are asynchronously written to the file system
  - When the file system structures are modified, the transaction is removed from the log
  - If the file system crashes, all remaining transactions in the log must still be performed
- Faster recovery from crash, removes chance of inconsistency of metadata

Next Lecture

We wrap up file systems