Disk and File Management

Categories of I/O Devices

- External devices that engage in I/O with computer systems can be grouped into three categories:

Human readable

- Suitable for communicating with the computer user
- Printers, terminals, video display, keyboard, mouse

Machine readable

- Suitable for communicating with electronic equipment
- Disk drives, USB keys, sensors, controllers

Communication

- Suitable for communicating with remote devices
- Modems, digital line drivers

Differences in I/O Devices

- Devices differ in a number of areas:

Data Rate

•There may be differences of magnitude between the data transfer rates

Application

The use to which a device is put has an influence on the software

Complexity of Control

•The effect on the operating system is filtered by the complexity of the I/O module that controls the device

Unit of Transfer

•Data may be transferred as a stream of bytes or characters or in larger blocks

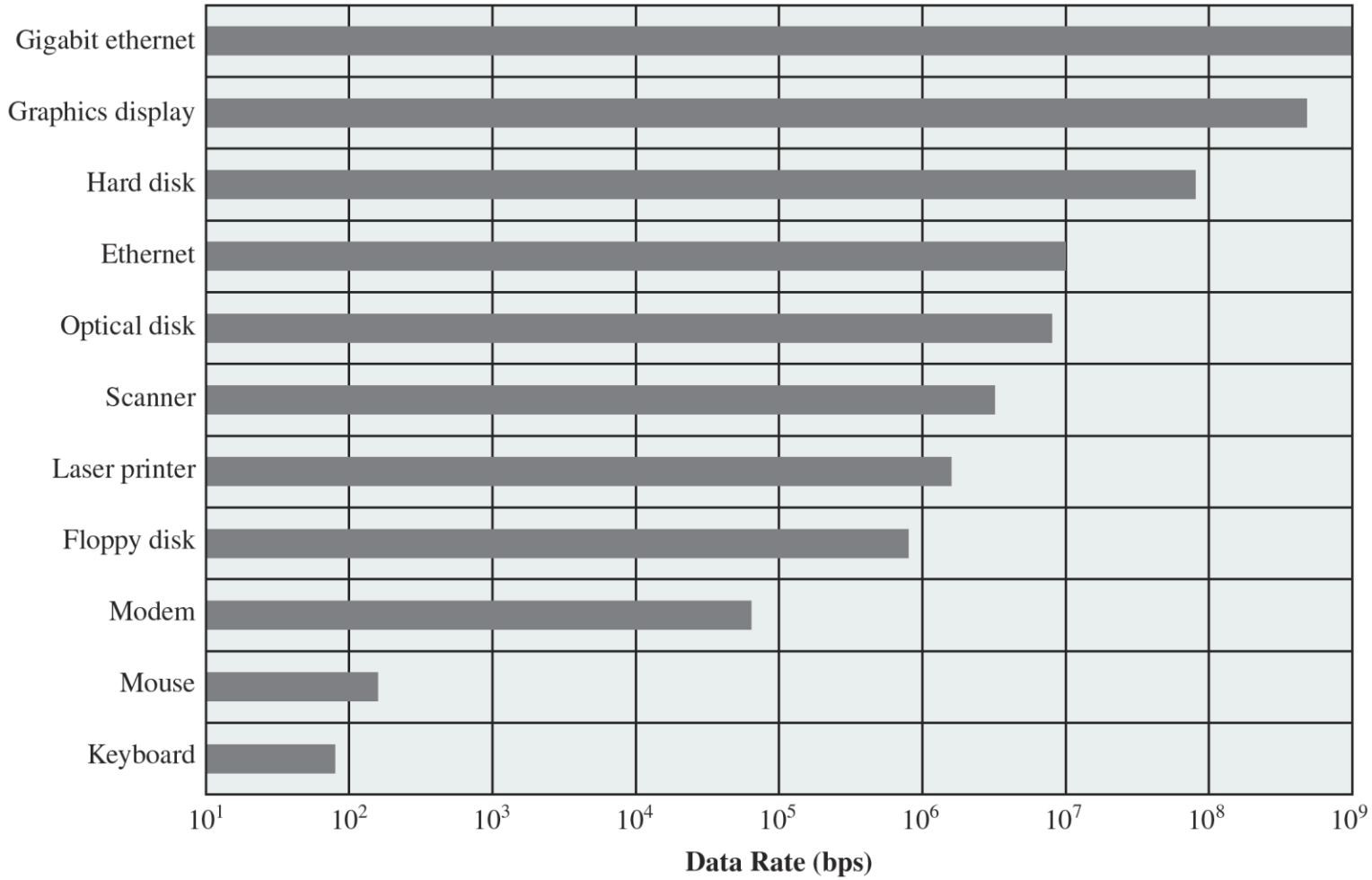
Data Representation

•Different data encoding schemes are used by different devices

Error Conditions

•The nature of errors, the way in which they are reported

Typical I/O Device Data Rates



Organization of the I/O Function

- Programmed I/O
  - The processor issues an I/O command on behalf of a process to an I/O module; that process then busy waits for the operation to be completed before proceeding
- Interrupt-driven I/O
  - The processor issues an I/O command on behalf of a process
    - If non-blocking – processor continues to execute instructions from the process that issued the I/O command
    - If blocking – the next instruction the processor executes is from the OS, which will put the current process in a blocked state and schedule another process
- Direct Memory Access (DMA)
  - A DMA module controls the exchange of data between main memory and an I/O module

Evolution of the I/O Function

1. Processor directly controls a peripheral device
2. A controller or I/O module is added
3. Same configuration as step 2, but now interrupts are employed
4. The I/O module is given direct control of memory via DMA
5. The I/O module is enhanced to become a separate processor, with a specialized instruction set tailored for I/O
6. The I/O module has a local memory of its own and is, in fact, a computer in its own right

Design Objectives

Efficiency

- Major effort in I/O design
- Important because I/O operations often form a bottleneck
- Most I/O devices are extremely slow compared with main memory and the processor
- The area that has received the most attention is disk I/O

Generality

- Desirable to handle all devices in a uniform manner
- Applies to the way processes view I/O devices and the way the operating system manages I/O devices and operations
- Diversity of devices makes it difficult to achieve true generality
- Use a hierarchical, modular approach to the design of the I/O function

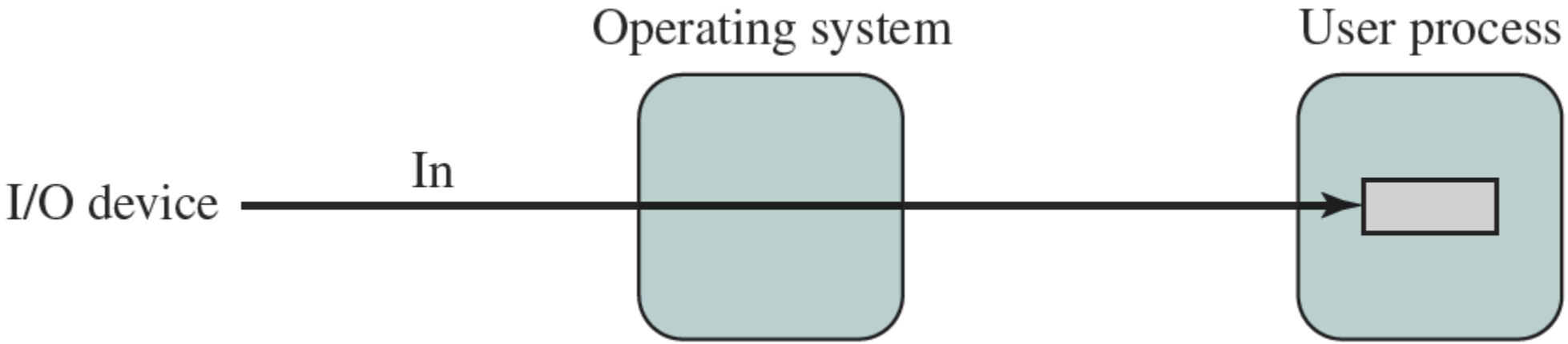
Buffering

- To avoid overheads and inefficiencies, it is sometimes convenient to perform input transfers in advance of requests being made, and to perform output transfers some time after the request is made

Block-oriented device	Stream-oriented device
• Stores information in blocks that are usually of fixed size • Transfers are made one block at a time • Possible to reference data by its block number • Disks and USB keys are examples	• Transfers data in and out as a stream of bytes • No block structure • Terminals, printers, communications ports, mouse and other pointing devices, and most other devices that are not secondary storage are examples

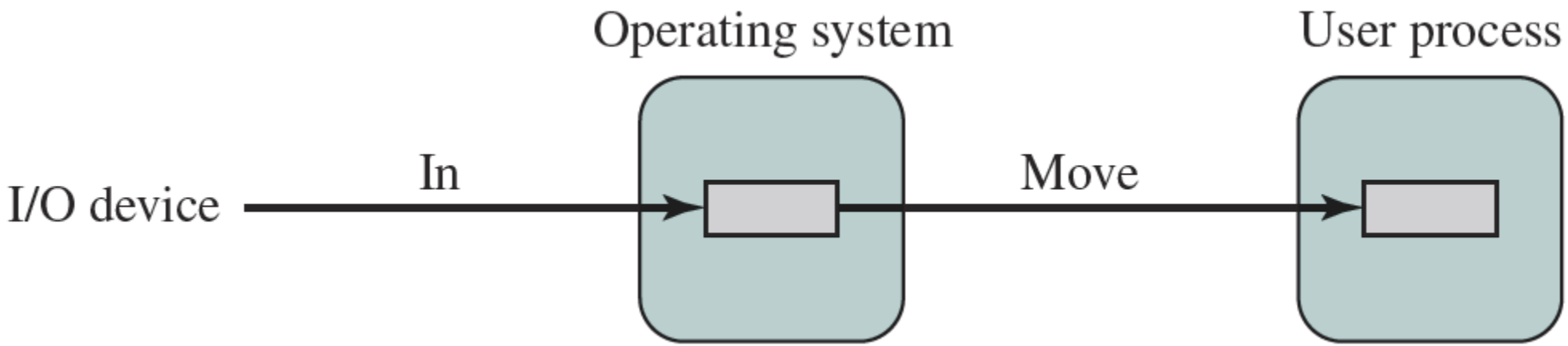
No Buffer

- Without a buffer, the OS directly accesses the device when it needs



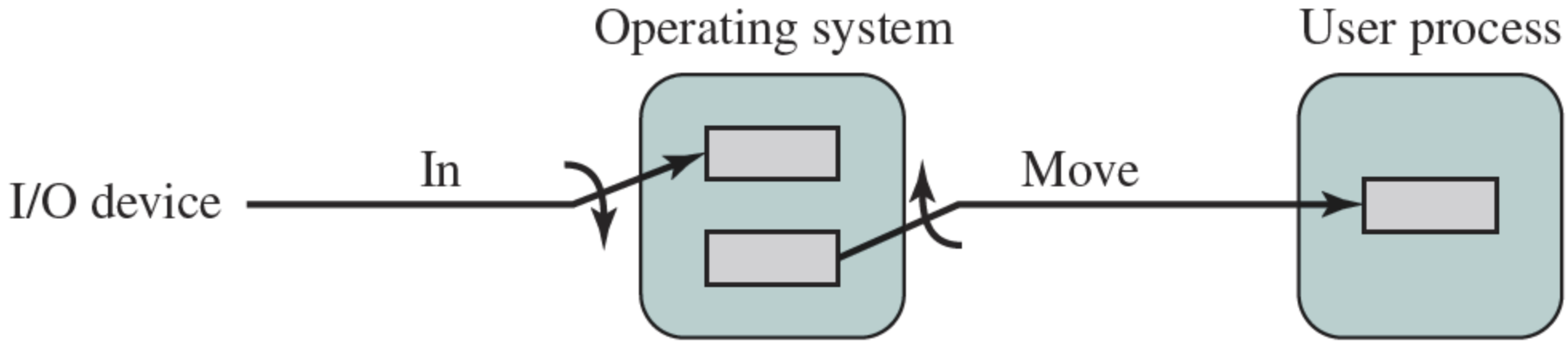
Single Buffer

- The simplest type of support that the operating system can provide
- When a user process issues an I/O request, the OS assigns a buffer in the system portion of main memory to the operation



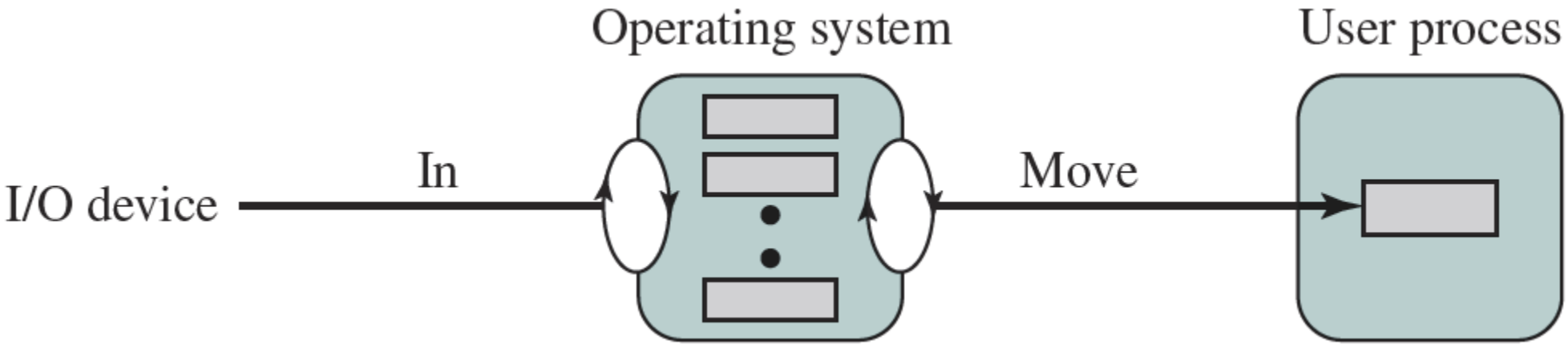
Double Buffer

- Assigning two system buffers to the operation
- A process now transfers data to or from one buffer while the operating system empties or fills the other buffer
- Also known as buffer swapping



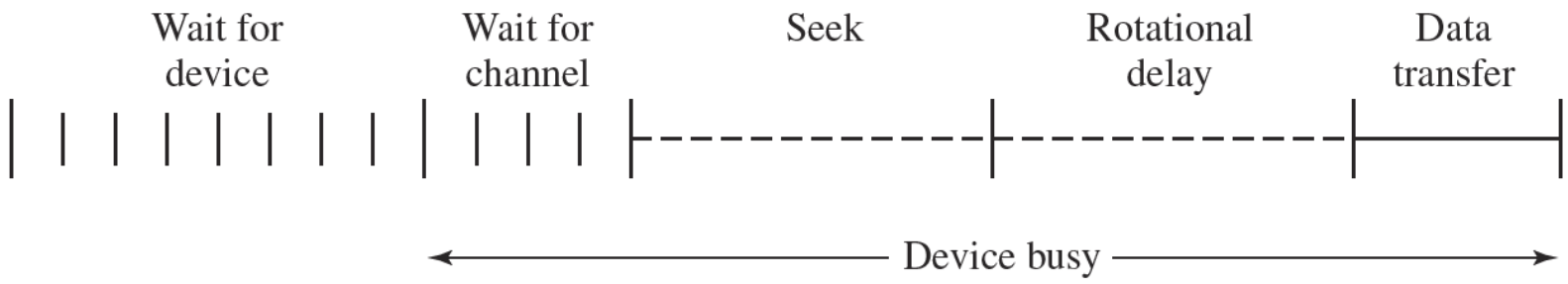
Circular Buffer

- When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer
- Each individual buffer is one unit in the circular buffer



Disk Performance Parameters

- The actual details of disk I/O operation depend on the:
  - Computer system
  - Operating system
  - Nature of the I/O channel and disk controller hardware



Disk Performance Parameters

- When the disk drive is operating, the disk is rotating at constant speed
- To read or write the head must be positioned at the desired track and at the beginning of the desired sector on that track
- Track selection involves moving the head in a movable-head system or electronically selecting one head on a fixed-head system
- On a movable-head system the time it takes to position the head at the track is known as seek time
- The time it takes for the beginning of the sector to reach the head is known as rotational delay
- The sum of the seek time and the rotational delay equals the access time

Seek Time

- The time required to move the disk arm to the required track
- Consists of two key components:
  - The initial startup time
  - The time taken to traverse the tracks that have to be crossed once the access arm is up to speed
- Settling time
  - Time after positioning the head over the target track until track identification is confirmed
- Much improvement comes from smaller and lighter disk components
- A typical average seek time on contemporary hard disks is under 10ms

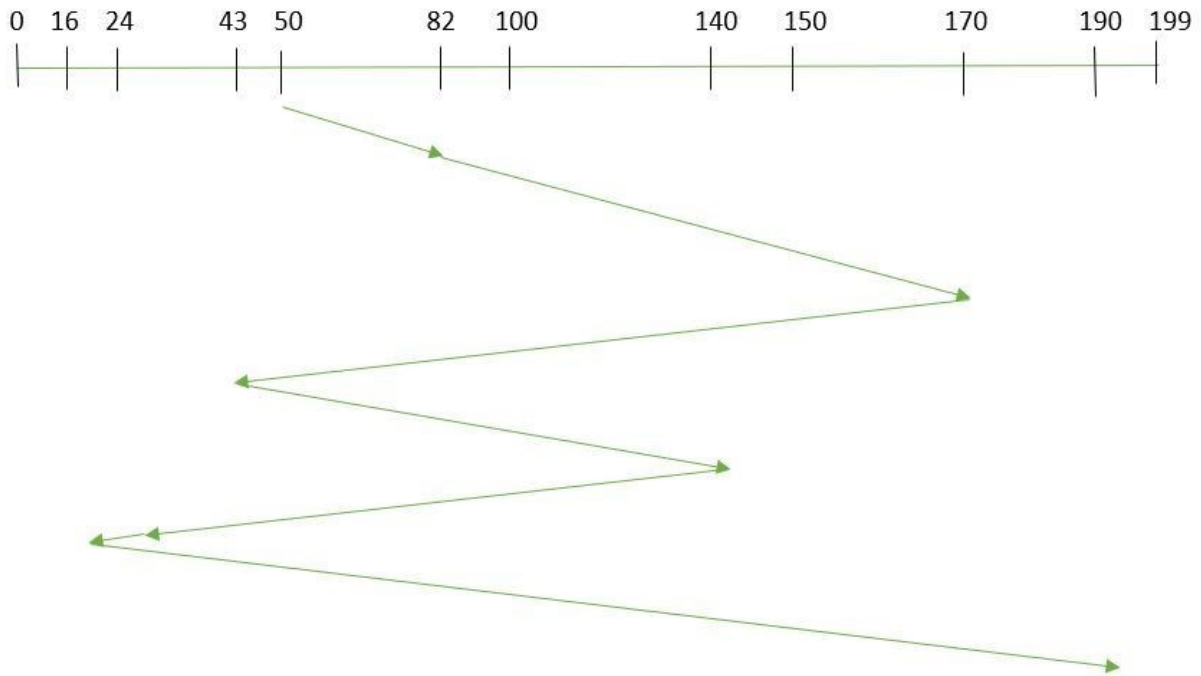
Disk Performance

- Rotational delay
  - The time required for the addressed area of the disk to rotate into a position where it is accessible by the read/write head
  - Disks rotate at speeds ranging from 3,6000 rpm (for handheld devices such as digital cameras) up to 15,000 rpm

First Come – First Serve (FCFS – FIFO)

- Processes in sequential order
- Fair to all processes
- Approximates random scheduling in performance if there are many processes competing for the disk

Current position of Read/Write head is: 50
Order of request is: 82,170, 43,140, 24,16,190



Total seek time =(82-50)+(170-82)+(170-43)+(140-43)+(140-24)+(24-16)+(190-16) = 642 | Average seek length = 91.7

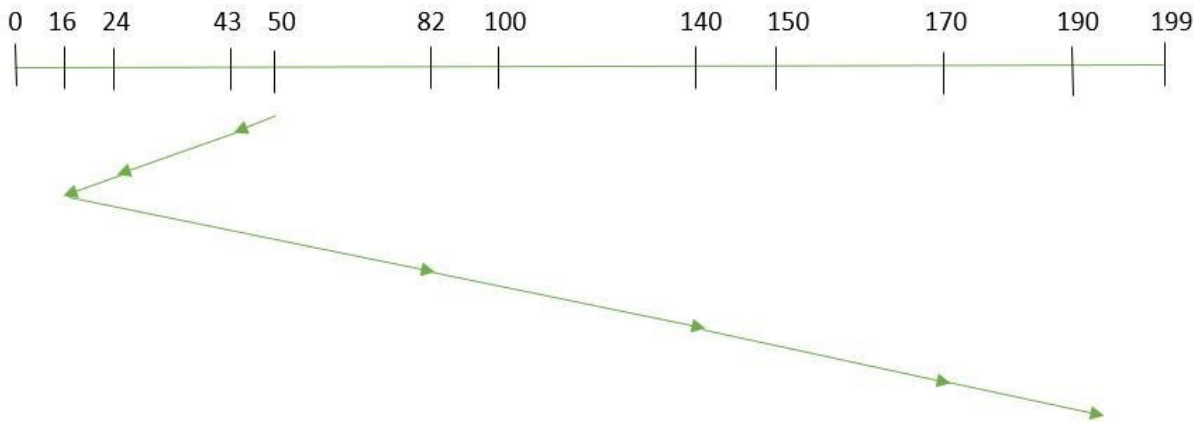
Priority (PRI)

- Control of the scheduling is outside the control of disk management software
- Goal is not to optimize disk utilization but to meet other objectives
- Short batch jobs and interactive jobs are given higher priority
- Provides good interactive response time
- Longer jobs may have to wait an excessively long time
- A poor policy for database systems

Shortest Seek Time First (SSTF)

- Select the disk I/O request that requires the least movement of the disk arm from its current position
- Always choose the minimum seek time
- If the seek time is same in both direction continues to move in the same direction

Current position of Read/Write head is: 50
Order of request is: 82,170,43,140,24,16,190

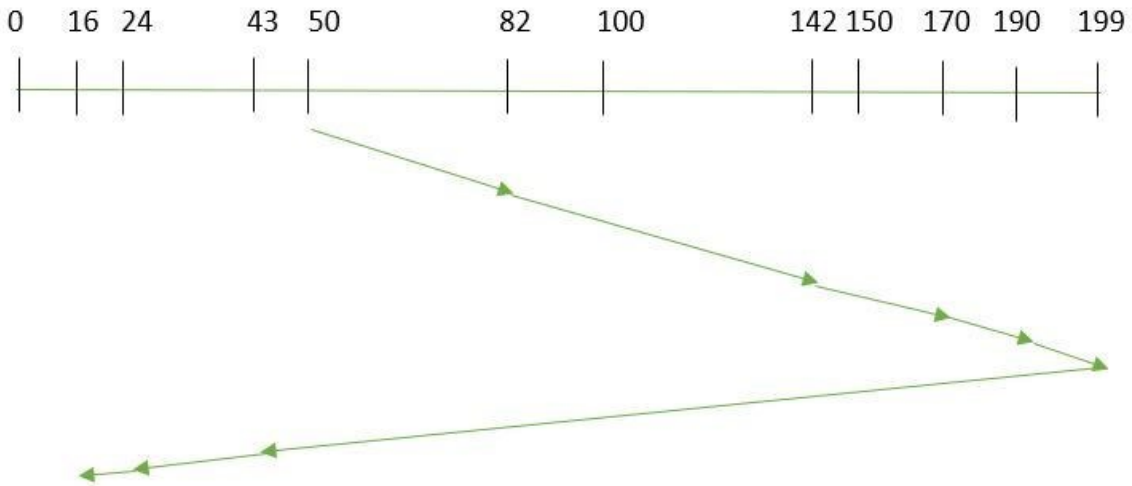


Total seek time = (50-43)+(43-24)+(24-16)+(82-16)+(140-82)+(170-140)+(190-170) = 208 | Average seek length = 29.7

SCAN

- Also known as the elevator algorithm
- Arm moves in one direction only
  - Satisfies all outstanding requests until it reaches the last track on disk then the direction is reversed
- Favors jobs whose requests are for tracks nearest to both innermost and outermost tracks and favors the latest-arriving jobs

Current position of Read/Write head is: 50 moving to end
Order of request is: 82,170,43,140,24,16,190

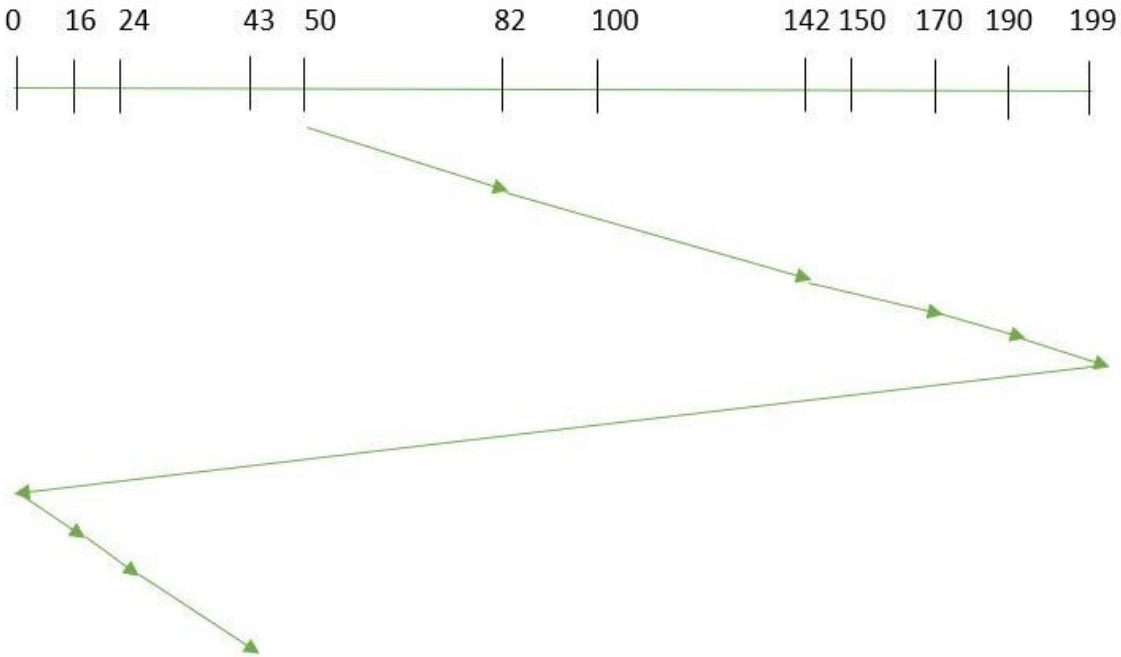


Total seek time = (199-50)+(199-16) = 332 | Average seek length = 47.4

C-SCAN (Circular SCAN)

- Restricts scanning to one direction only
- When the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again

Current position of Read/Write head is: 50 moving to end
Order of request is: 82,170,43,140,24,16,190

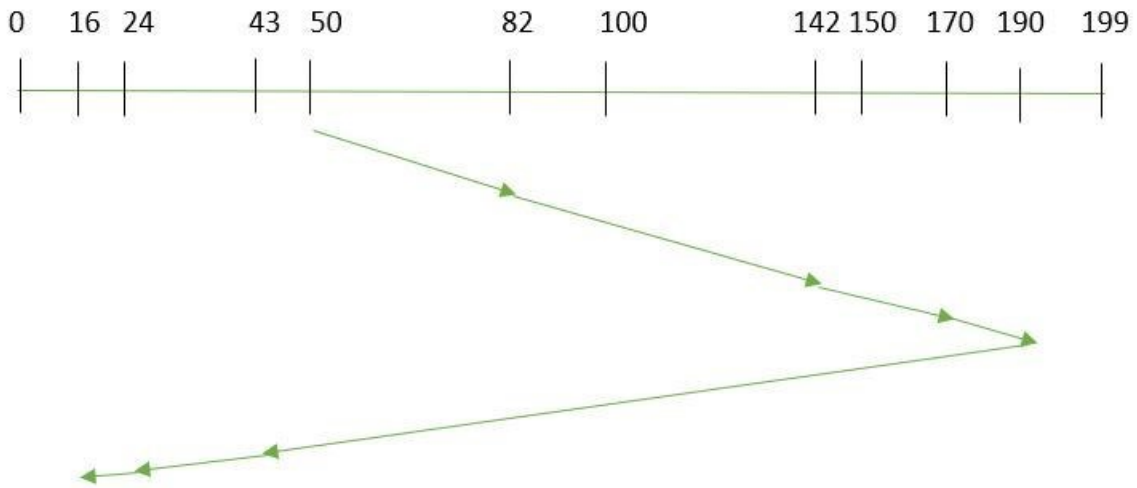


Total seek time = (199-50)+(199-0)+(43-0) = 391 | Average seek length = 55.8

LOOK

- It is similar to the SCAN disk scheduling algorithm except for the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only

Current position of Read/Write head is: 50 moving to end
Order of request is: 82,170,43,140,24,16,190

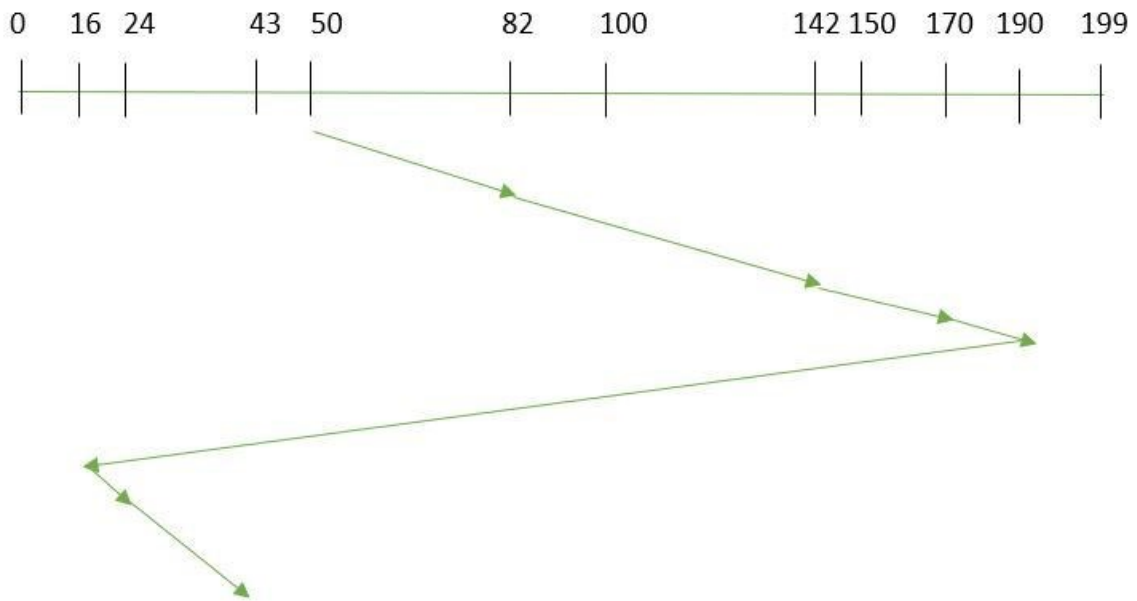


Total seek time = (190-50)+(190-16) = 314 | Average seek length = 44.8

C-LOOK

- C-LOOK is similar to C-SCAN disk scheduling algorithm.
- In CLOOK, the disk arm in spite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request.

Current position of Read/Write head is: 50 moving to end
Order of request is: 82,170,43,140,24,16,190



Total seek time = (190-50)+(190-16)+(43-16) = 341 | Average seek length = 48.7

N-Step-SCAN

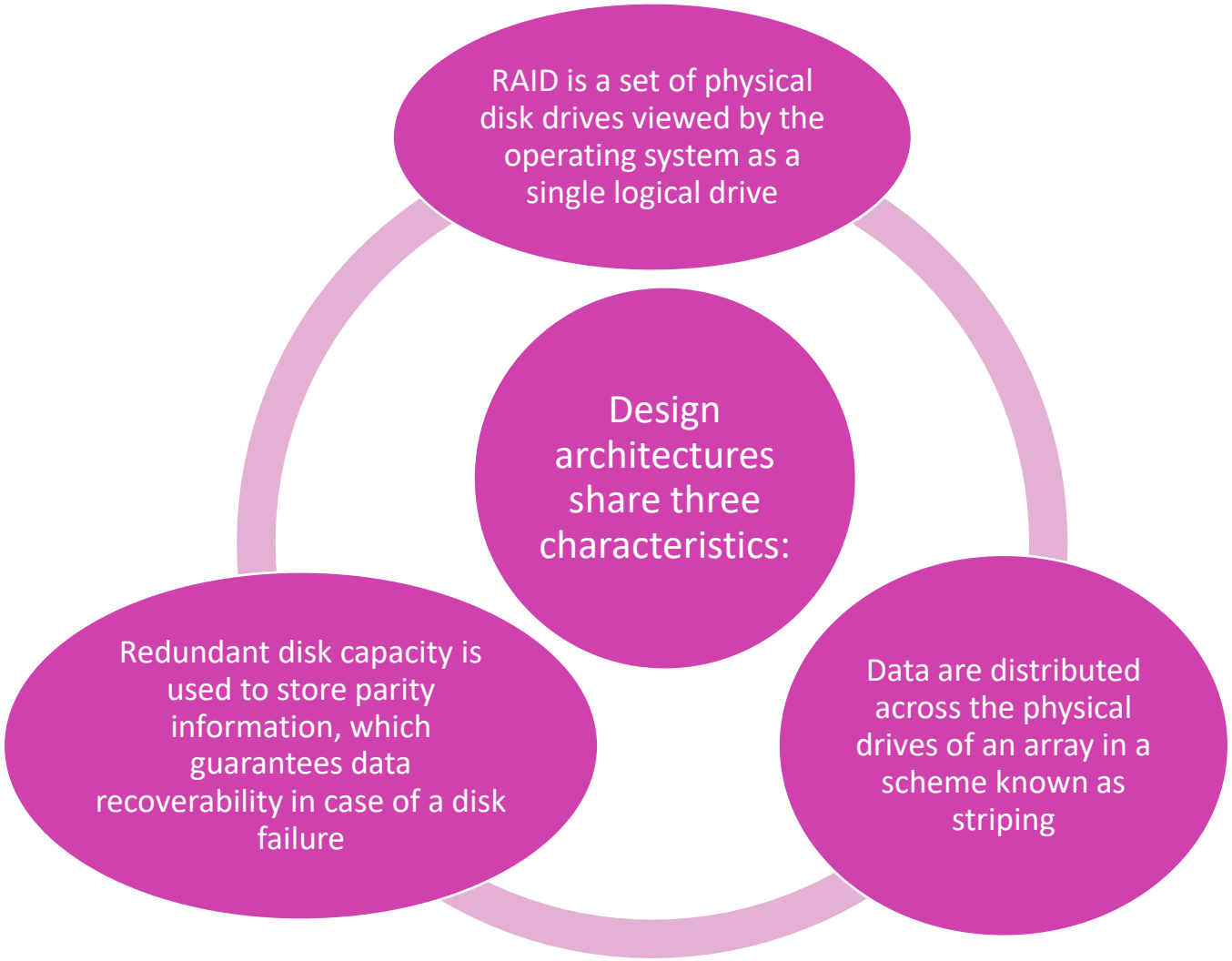
- Segments the disk request queue into subqueues of length N
- Subqueues are processed one at a time, using SCAN
- While a queue is being processed new requests must be added to some other queue
- If fewer than N requests are available at the end of a scan, all of them are processed with the next scan

FSCAN

- Uses two subqueues
- When a scan begins, all of the requests are in one of the queues, with the other empty
- During scan, all new requests are put into the other queue
- Service of new requests is deferred until all of the old requests have been processed

RAID

- Redundant Array of Independent Disks
- Consists of seven levels, zero through six

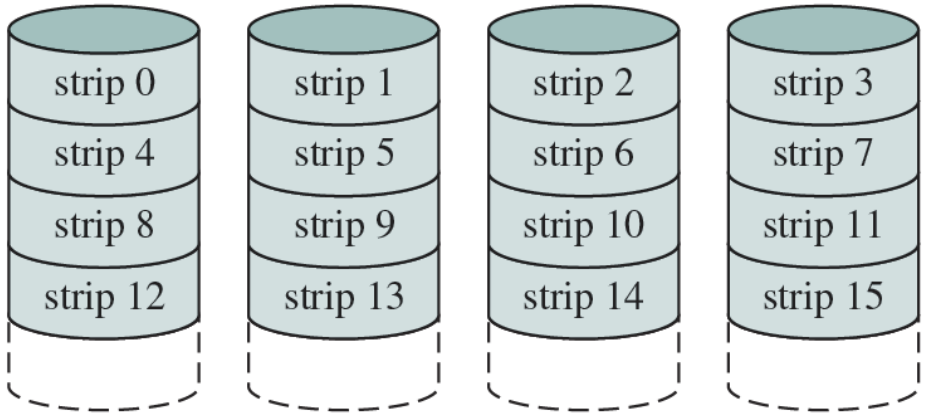


RAID

- The term was originally coined in a paper by a group of researchers at the University of California at Berkeley
  - The paper outlined various configurations and applications and introduced the definitions of the RAID levels
- Strategy employs multiple disk drives and distributes data in such a way as to enable simultaneous access to data from multiple drives
  - Improves I/O performance and allows easier incremental increases in capacity
- The unique contribution is to address effectively the need for redundancy
- Makes use of stored parity information that enables the recovery of data lost due to a disk failure

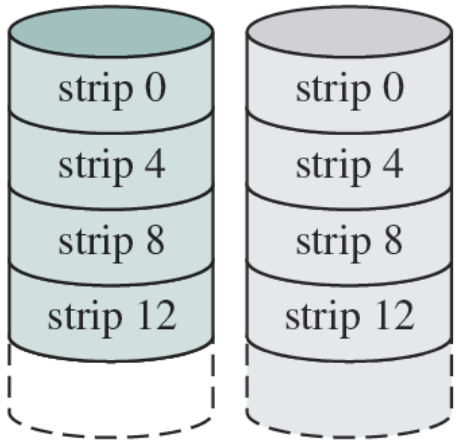
RAID Level 0

- Not a true RAID because it does not include redundancy to improve performance or provide data protection
- User and system data are distributed across all of the disks in the array
- Logical disk is divided into strips



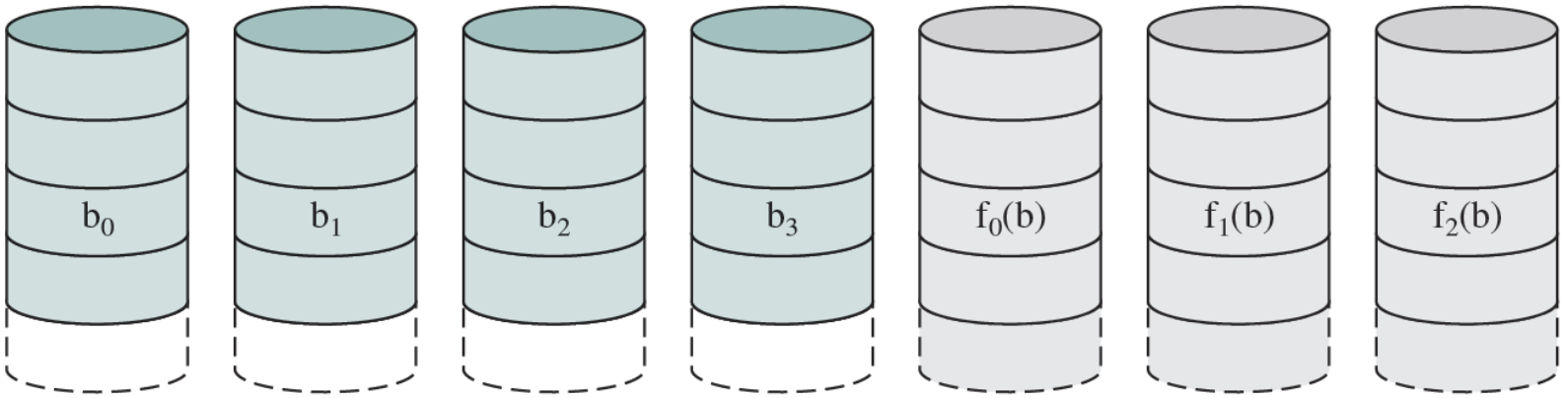
RAID Level 1

- Redundancy is achieved by the simple expedient of duplicating all the data
- There is no “write penalty”
- When a drive fails the data may still be accessed from the second drive
- Principal disadvantage is the cost



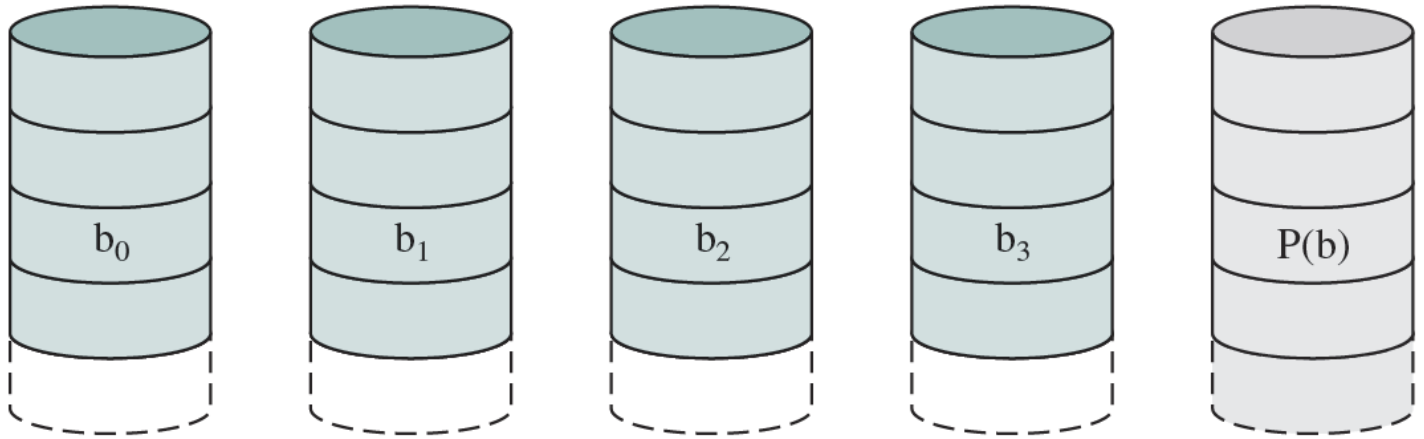
RAID Level 2

- Makes use of a parallel access technique
- Data striping is used
- Typically a Hamming code is used
- Effective choice in an environment in which many disk errors occur



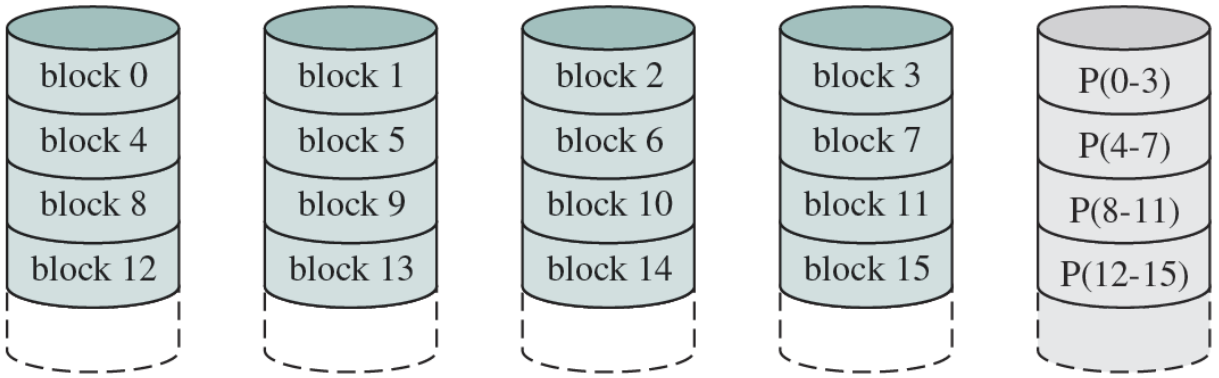
RAID Level 3

- Requires only a single redundant disk, no matter how large the disk array
- Employs parallel access, with data distributed in small strips
- Can achieve very high data transfer rates



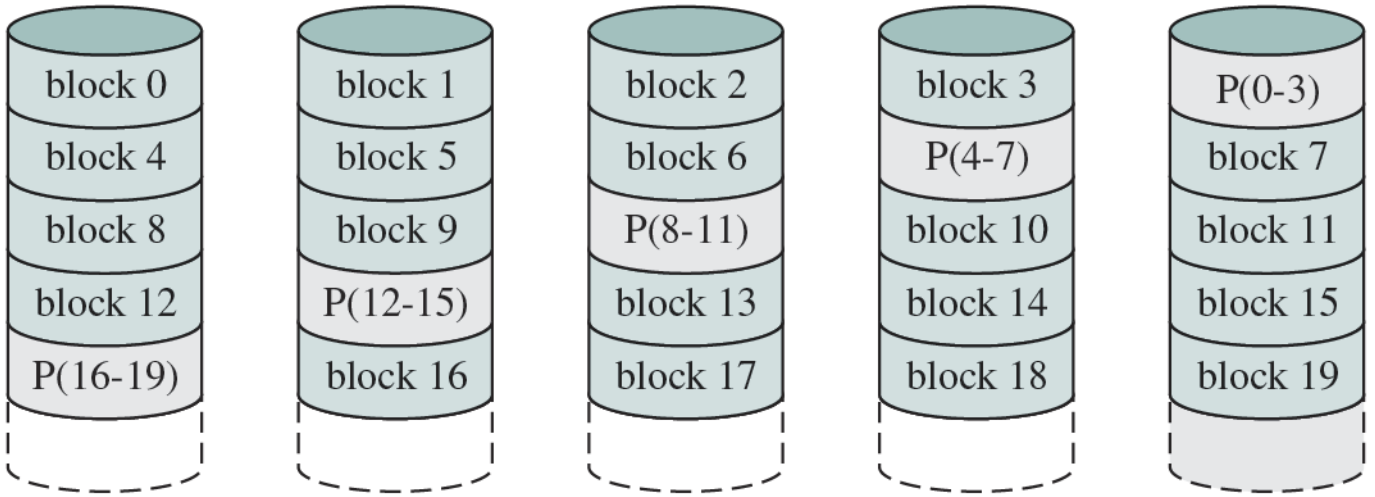
RAID Level 4

- Makes use of an independent access technique
- A bit-by-bit parity strip is calculated across corresponding strips on each data disk, and the parity bits are stored in the corresponding strip on the parity disk
- Involves a write penalty when an I/O write request of small size is performed



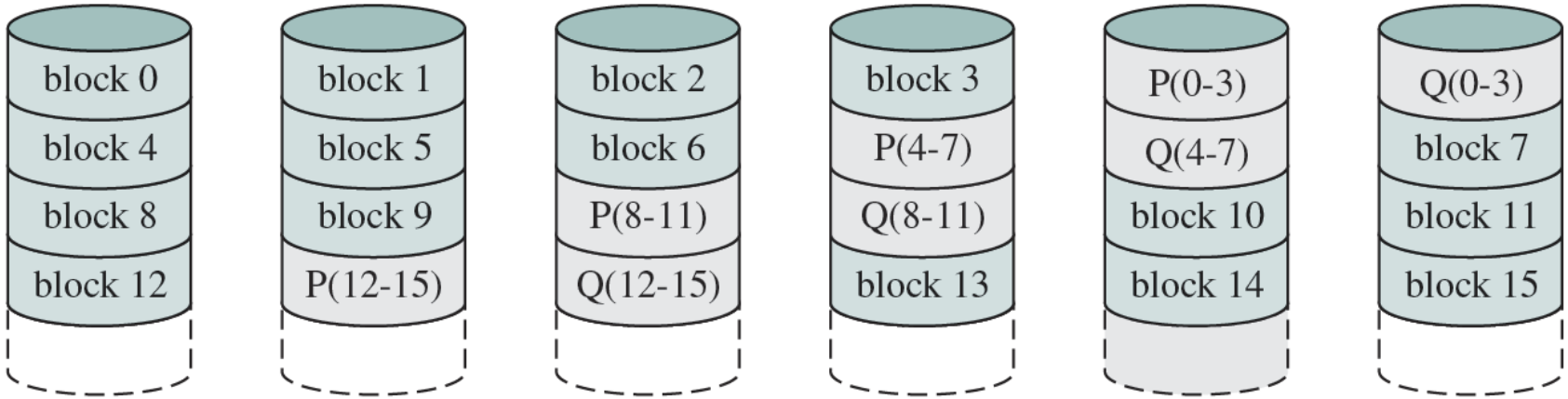
RAID Level 5

- Similar to RAID-4 but distributes the parity bits across all disks
- Typical allocation is a round-robin scheme
- Has the characteristic that the loss of any one disk does not result in data loss



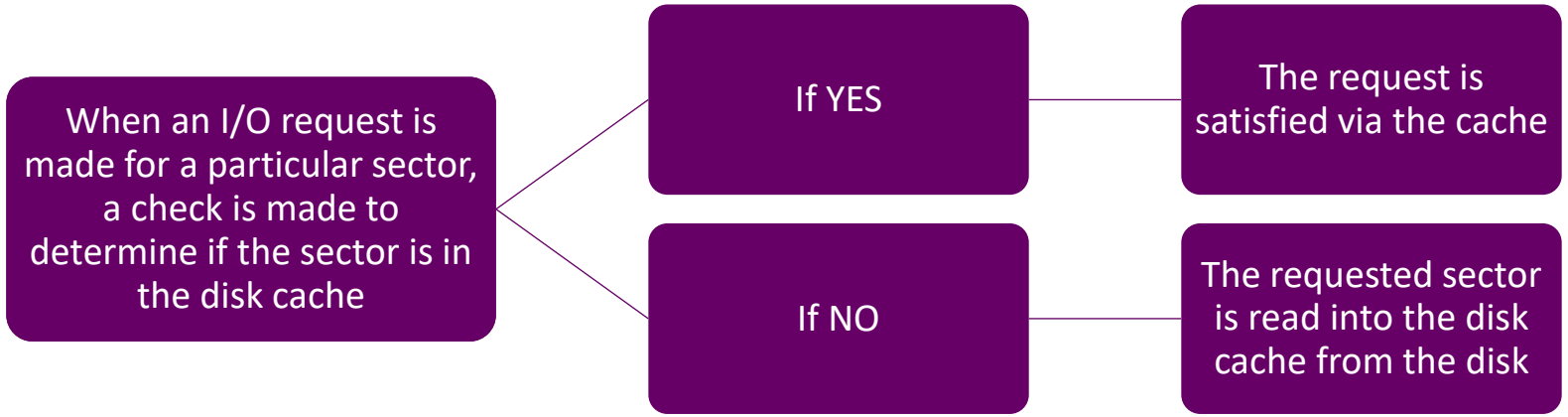
RAID Level 6

- Two different parity calculations are carried out and stored in separate blocks on different disks
- Provides extremely high data availability
- Incurs a substantial write penalty because each write affects two parity blocks



Disk Cache

- Cache memory is used to apply to a memory that is smaller and faster than main memory and that is interposed between main memory and the processor
- Reduces average memory access time by exploiting the principle of locality
- Disk cache is a buffer in main memory for disk sectors
- Contains a copy of some of the sectors on the disk



Least Recently Used (LRU)

- Most commonly used algorithm that deals with the design issue of replacement strategy
- The block that has been in the cache the longest with no reference to it is replaced
- A stack of pointers reference the cache
  - Most recently referenced block is on the top of the stack
  - When a block is referenced or brought into the cache, it is placed on the top of the stack

Least Frequently Used (LFU)

- The block that has experienced the fewest references is replaced
- A counter is associated with each block
- Counter is incremented each time block is accessed
- When replacement is required, the block with the smallest count is selected

UNIX Buffer Cache

- Is essentially a disk cache
  - I/O operations with disk are handled through the buffer cache
- The data transfer between the buffer cache and the user process space always occurs using DMA
  - Does not use up any processor cycles
  - Does consume bus cycles
- Three lists are maintained:
  - Free list
    - List of all slots in the cache that are available for allocation
  - Device list
    - List of all buffers currently associated with each disk
  - Driver I/O queue
    - List of buffers that are actually undergoing or waiting for I/O on a particular device

Character Queue

Used by character oriented devices

Terminals and printers

Either written by the I/O device and read by the process or vice versa

Producer/consumer model is used

Character queues may only be read once

As each character is read, it is effectively destroyed

Unbuffered I/O

- Is simply DMA between device and process space
- Is always the fastest method for a process to perform I/O
- Process is locked in main memory and cannot be swapped out
- I/O device is tied up with the process for the duration of the transfer making it unavailable for other processes

Linux I/O

- Very similar to other UNIX implementation
- Associates a special file with each I/O device driver
- Block, character, and network devices are recognized
- Default disk scheduler in Linux 2.4 is the Linux Elevator

For Linux 2.6 the Elevator algorithm has been augmented by two additional algorithms:

- The deadline I/O scheduler
- The anticipatory I/O scheduler

The Elevator Scheduler

- Maintains a single queue for disk read and write requests and performs both sorting and merging functions on the queue
- When a new request is added to the queue, four operations are considered in order:
  - If the request is to the same on-disk sector or an immediately adjacent sector to a pending request in the queue, then the existing request and the new request are merged into one request
  - If a request in the queue is sufficiently old, the new request is inserted at the tail of the queue
  - If there is a suitable location, the new request is inserted in sorted order
  - If there is no suitable location, the new request is placed at the tail of the queue

Deadline Scheduler

- Two problems manifest themselves with the elevator scheme:
  - A distant block request can be delayed for a substantial time because the queue is dynamically updated
  - A stream of write requests can block a read request for a considerable time, and thus block a process
- Deadline I/O scheduler was developed in 2002
  - This scheduler makes use of two pairs of queues
  - In addition to each incoming request being placed in a sorted elevator queue as before, the same request is placed at the tail of a read FIFO queue for a read request or a write FIFO queue for a write request
  - When a request is satisfied, it is removed from the head of the sorted queue and also from the appropriate FIFO queue

Anticipatory I/O Scheduler

- Elevator and deadline scheduling can be counterproductive if there are numerous synchronous read requests
- In Linux, the anticipatory scheduler is superimposed on the deadline scheduler
- When a read request is dispatched, the anticipatory scheduler causes the scheduling system to delay
  - There is a good chance that the application that issued the last read request will issue another read request to the same region of the disk
    - That request will be serviced immediately
    - Otherwise the scheduler resumes using the deadline scheduling algorithm

The NOOP Scheduler

- This is the simplest among Linux I/O schedulers
- It is a minimal scheduler that inserts I/O requests into a FIFO queue and uses merging
- Its main uses include nondisk-based block devices such as memory devices, and specialized software or hardware environments that do their own scheduling and need only minimal support in the kernel

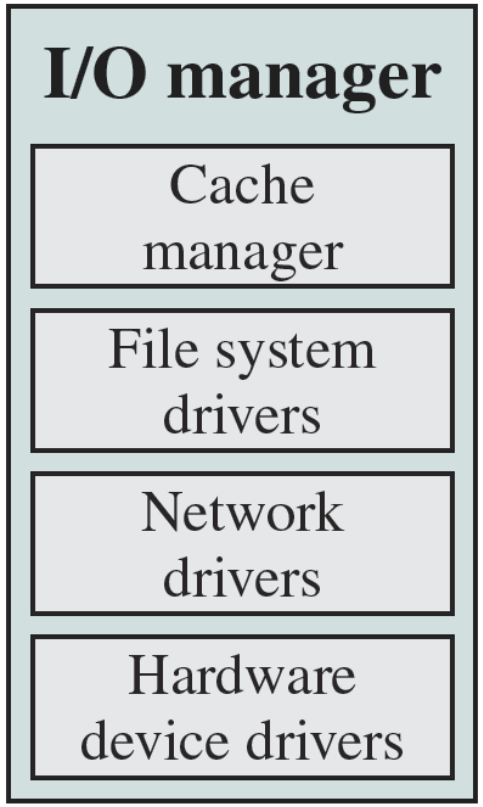
Completely Fair Queuing I/O Scheduler (CFQ)

- Was developed in 2003
- Is the default I/O scheduler in Linux
- The CFQ scheduler guarantees a fair allocation of the disk I/O bandwidth among all processes
- It maintains per process I/O queues
  - Each process is assigned a single queue
  - Each queue has an allocated timeslice
  - Requests are submitted into these queues and are processed in round robin
- When the scheduler services a specific queue, and there are no more requests in that queue, it waits in idle mode for a predefined time interval for new requests, and if there are no requests, it continues to the next queue

Linux Page Cache

- For Linux 2.4 and later there is a single unified page cache for all traffic between disk and main memory
- Benefits:
  - When it is time to write back dirty pages to disk, a collection of them can be ordered properly and written out efficiently
  - Because of the principle of temporal locality, pages in the page cache are likely to be referenced again before they are flushed from the cache, thus saving a disk I/O operation

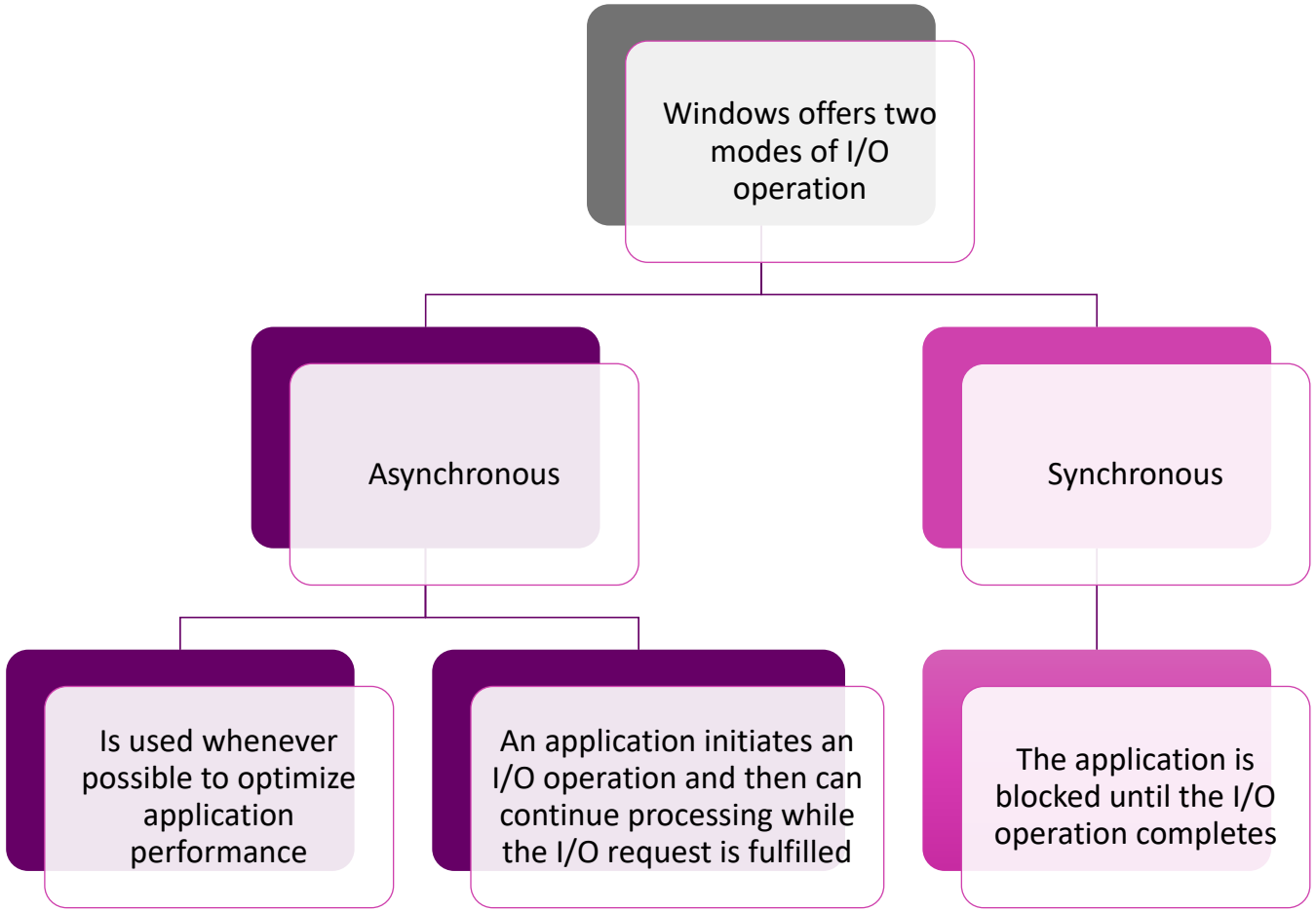
Windows I/O Manager



Basic I/O Facilities

- Cache Manager
  - Maps regions of files into kernel virtual memory and then relies on the virtual memory manager to copy pages to and from the files on disk
- File System Drivers
  - Sends I/O requests to the software drivers that manage the hardware device adapter
- Network Drivers
  - Windows includes integrated networking capabilities and support for remote file systems
  - The facilities are implemented as software drivers
- Hardware Device Drivers
  - The source code of Windows device drivers is portable across different processor types

Asynchronous and Synchronous I/O



I/O Completion

- Windows provides five different techniques for signaling I/O completion:

1. Signaling the file object
2. Signaling an event object
3. Asynchronous procedure call
4. I/O completion ports
5. Polling

Windows RAID Configurations

- Windows supports two sorts of RAID configurations:

Hardware RAID	Software RAID
Separate physical disks combined into one or more logical disks by the disk controller or disk storage cabinet hardware	Noncontiguous disk space combined into one or more logical partitions by the fault-tolerant software disk driver, FTDISK

Volume Shadow Copies and Volume Encryption

- Volume Shadow Copies
  - Efficient way of making consistent snapshots of volumes so they can be backed up
  - Also useful for archiving files on a per-volume basis
  - Implemented by a software driver that makes copies of data on the volume before it is overwritten
- Volume Encryption
  - Windows uses BitLocker to encrypt entire volumes
  - More secure than encrypting individual files
  - Allows multiple interlocking layers of security

Files

- Data collections created by users
- The File System is one of the most important parts of the OS to a user
- Desirable properties of files:

Long-term existence

- Files are stored on disk or other secondary storage and do not disappear when a user logs off

Sharable between processes

- Files have names and can have associated access permissions that permit controlled sharing

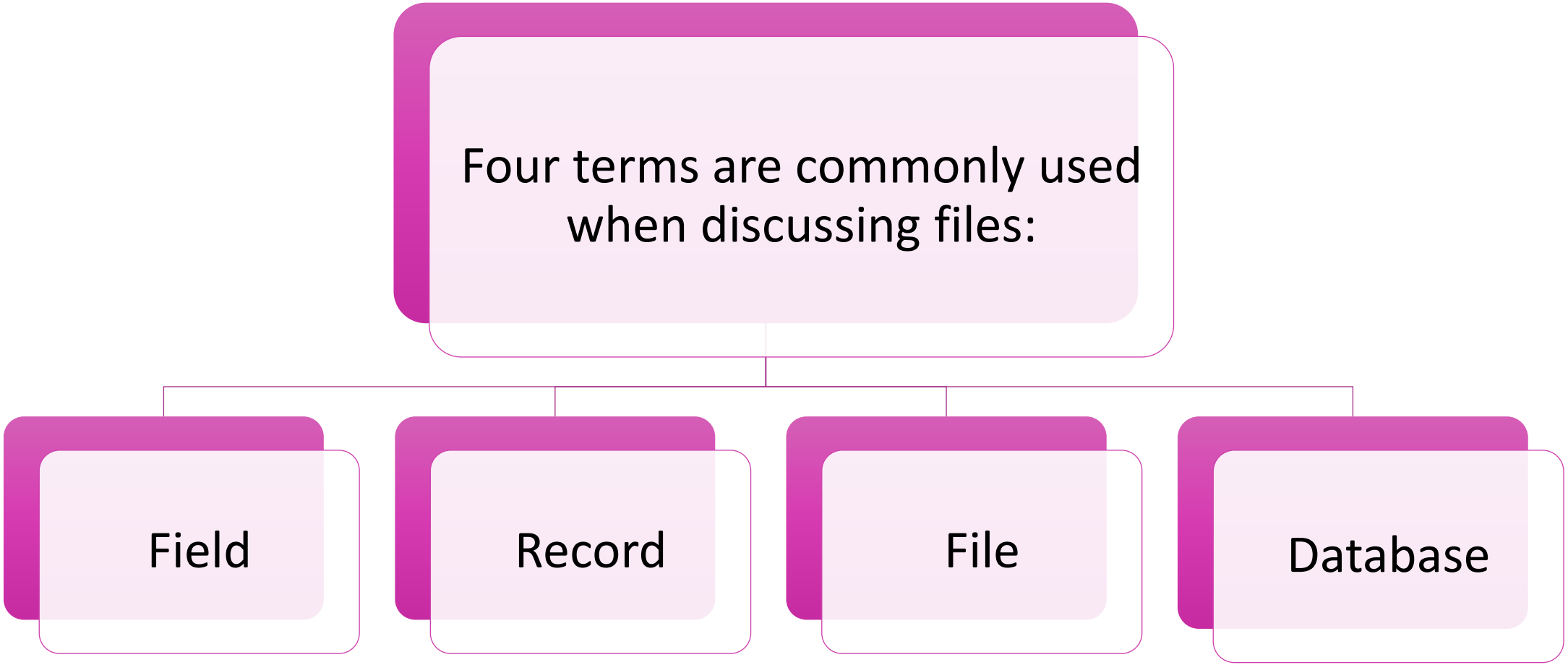
Structure

- Files can be organized into hierarchical or more complex structure to reflect the relationships among files

File Systems

- Provide a means to store data organized as files as well as a collection of functions that can be performed on files
- Maintain a set of attributes associated with the file
- Typical operations include:
  - Create
  - Delete
  - Open
  - Close
  - Read
  - Write

File Structure



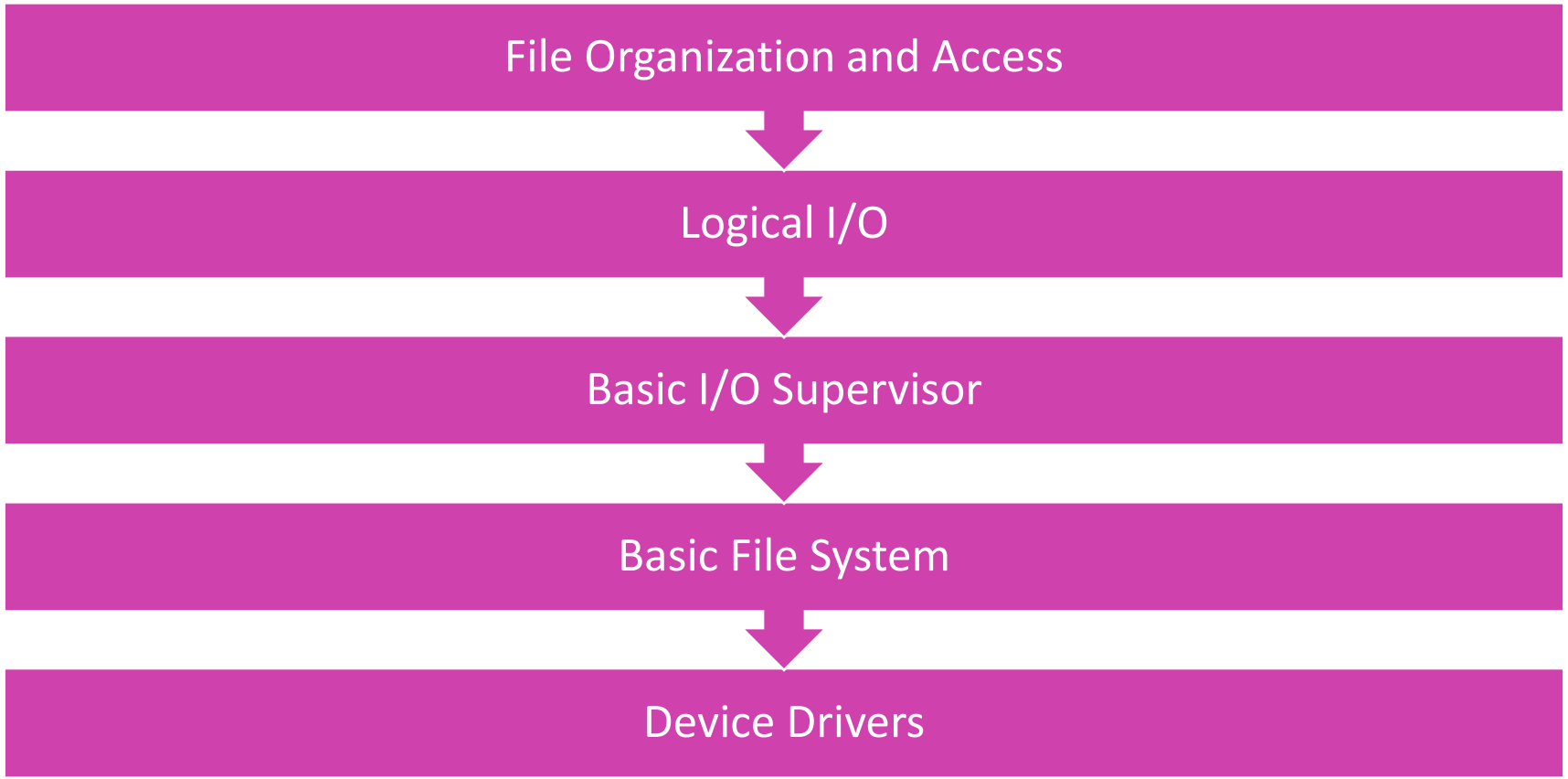
Structure Terms

- Field
  - Basic element of data
  - Contains a single value
  - Fixed or variable length
- Database
  - Collection of related data
  - Relationships among elements of data are explicit
  - Designed for use by a number of different applications
  - Consists of one or more types of files
- File
  - Collection of similar records
  - Treated as a single entity
  - May be referenced by name
  - Access control restrictions usually apply at the file level
- Record
  - Collection of related fields that can be treated as a unit by some application program
  - Fixed or variable length

File Management System

- Meet the data management needs of the user
- Guarantee that the data in the file are valid
- Optimize performance
- Provide I/O support for a variety of storage device types
- Minimize the potential for lost or destroyed data
- Provide a standardized set of I/O interface routines to user processes
- Provide I/O support for multiple users in the case of multiple-user systems

File System Software Architecture



Device Drivers

- Lowest level
- Communicates directly with peripheral devices
- Responsible for starting I/O operations on a device
- Processes the completion of an I/O request
- Considered to be part of the operating system

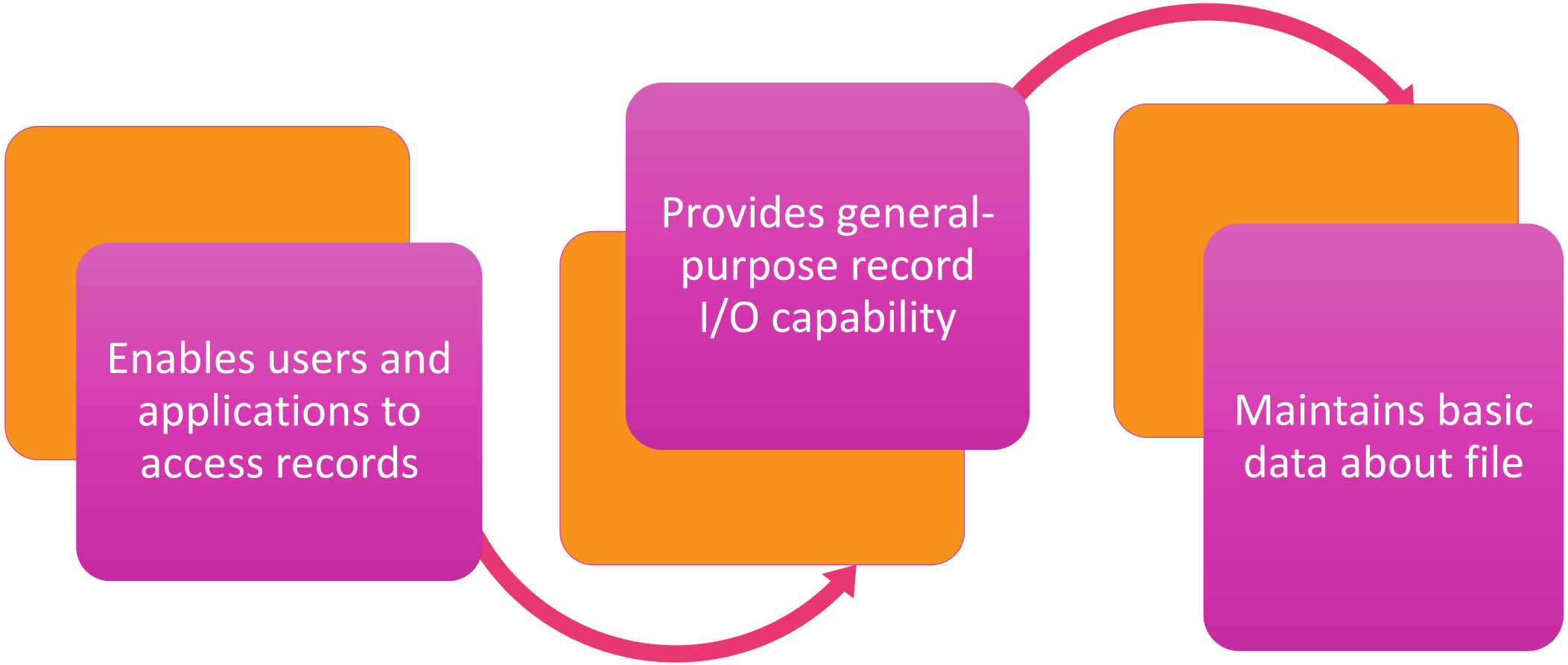
Basic File System

- Also referred to as the physical I/O level
- Primary interface with the environment outside the computer system
- Deals with blocks of data that are exchanged with disk or tape systems
- Concerned with the placement of blocks on the secondary storage device
- Concerned with buffering blocks in main memory
- Does not understand the content of the data or the structure of the files involved
- Considered part of the operating system

Basic I/O Supervisor

- Responsible for all file I/O initiation and termination
- At this level, control structures are maintained that deal with device I/O, scheduling, and file status
- Selects the device on which I/O is to be performed
- Concerned with scheduling disk and tape accesses to optimize performance
- I/O buffers are assigned and secondary memory is allocated at this level
- Part of the operating system

Logical I/O


Enables users and applications to access records
Provides general-purpose record I/O capability
Maintains basic data about file

Access Method

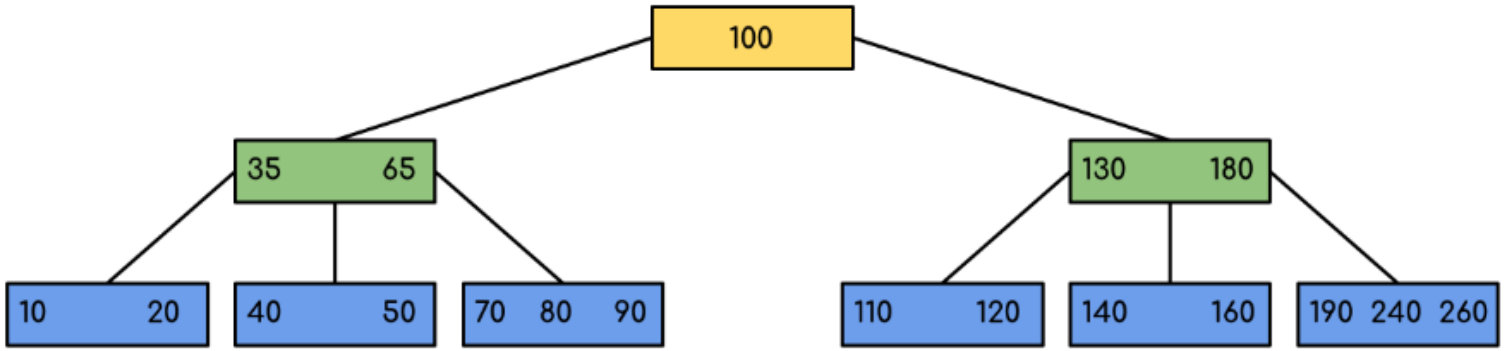
- Level of the file system closest to the user
- Provides a standard interface between applications and the file systems and devices that hold the data
- Different access methods reflect different file structures and different ways of accessing and processing the data

File Organization and Access

- File organization is the logical structuring of the records as determined by the way in which they are accessed
- In choosing a file organization, several criteria are important:
  - Short access time
  - Ease of update
  - Economy of storage
  - Simple maintenance
  - Reliability
- Priority of criteria depends on the application that will use the file

B-Trees

- A balanced tree structure with all branches of equal length
- Standard method of organizing indexes for databases
- Commonly used in OS file systems
- Provides for efficient searching, adding, and deleting of items

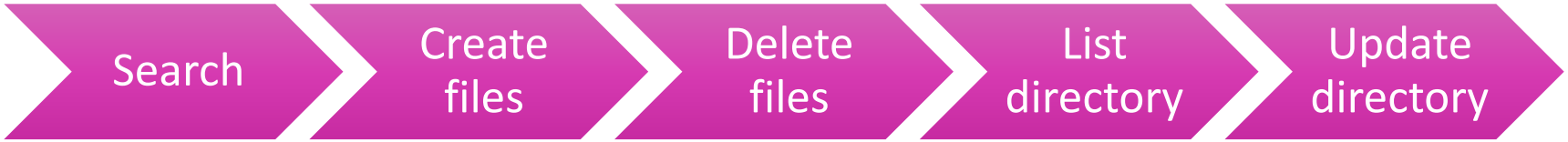


B-Tree Characteristics

- A B-tree is characterized by its minimum degree d and satisfies the following properties:
  1. Every node has at most 2d – 1 keys and 2d children or, equivalently, 2d pointers
  2. Every node, except for the root, has at least d – 1 keys and d pointers, as a result, each internal node, except the root, is at least half full and has at least d children
  3. The root has at least 1 key and 2 children
  4. All leaves appear on the same level and contain no information. This is a logical construct to terminate the tree; the actual implementation may differ
  5. A nonleaf node with k pointers contains k – 1 keys

Operations Performed on a Directory

- To understand the requirements for a file structure, it is helpful to consider the types of operations that may be performed on the directory:



Two-Level Scheme

There is one directory for each user and a master directory

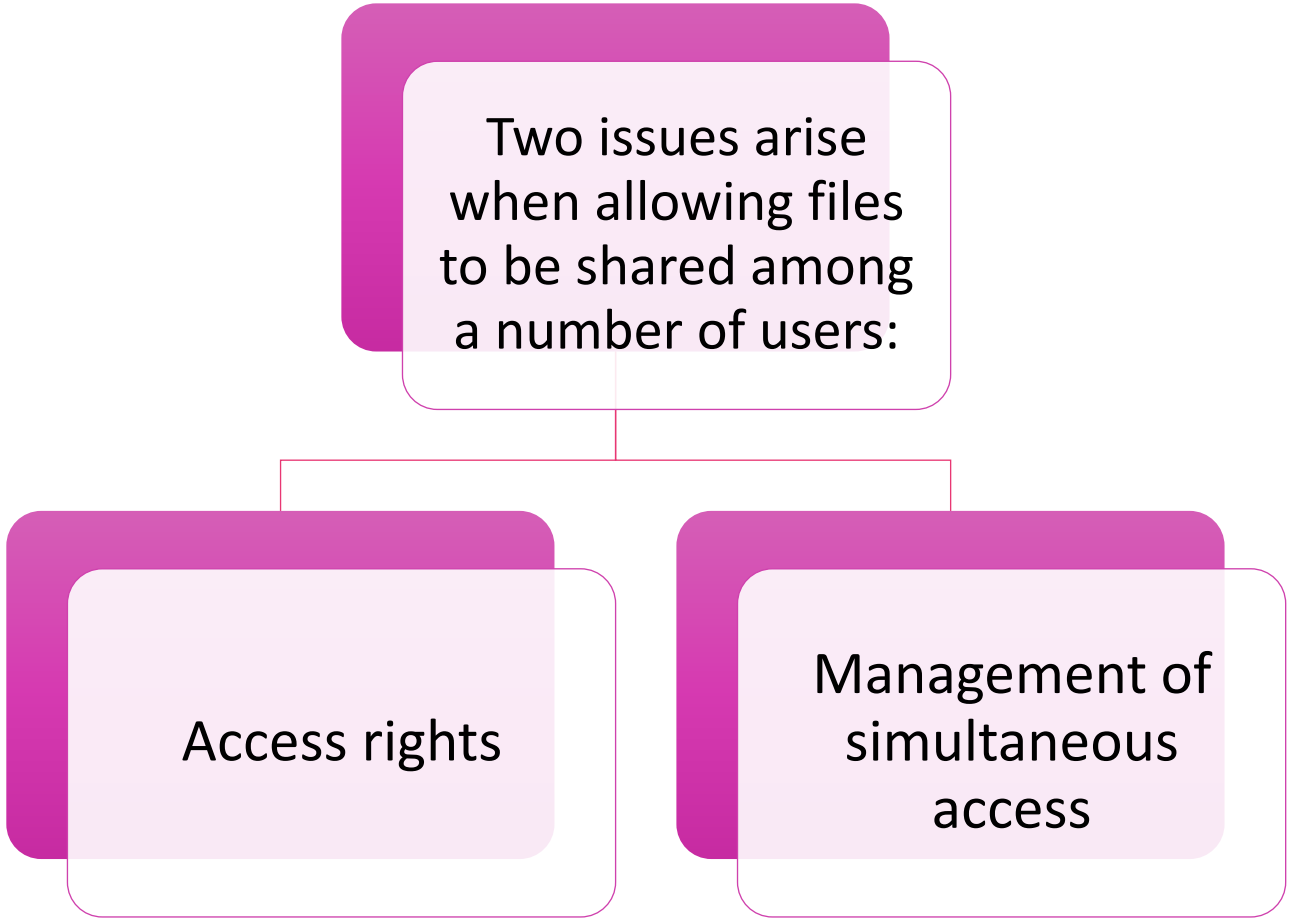
Master directory has an entry for each user directory providing address and access control information

Each user directory is a simple list of the files of that user

Names must be unique only within the collection of files of a single user

File system can easily enforce access restriction on directories

File Sharing



Access Rights

- None
  - The user would not be allowed to read the user directory that includes the file
- Knowledge
  - The user can determine that the file exists and who its owner is and can then petition the owner for additional access rights
- Execution
  - The user can load and execute a program but cannot copy it
- Reading
  - The user can read the file for any purpose, including copying and execution
- Appending
  - The user can add data to the file but cannot modify or delete any of the file's contents
- Updating
  - The user can modify, delete, and add to the file's data
- Changing protection
  - The user can change the access rights granted to other users
- Deletion
  - The user can delete the file from the file system

User Access Rights

Owner

- Usually the initial creator of the file
- Has full rights
- May grant rights to others

Specific Users

Individual users who are designated by user ID

User Groups

A set of users who are not individually defined

All

- All users who have access to this system
- These are public files

File Allocation

- On secondary storage, a file consists of a collection of blocks
- The operating system or file management system is responsible for allocating blocks to files
- The approach taken for file allocation may influence the approach taken for free space management
- Space is allocated to a file as one or more portions (contiguous set of allocated blocks)
- File allocation table (FAT)
    - Data structure used to keep track of the portions assigned to a file

Portion Size

- In choosing a portion size there is a trade-off between efficiency from the point of view of a single file versus overall system efficiency
- Items to be considered:
  1. Contiguity of space increases performance, especially for Retrieve_Next operations, and greatly for transactions running in a transaction-oriented operating system
  2. Having a large number of small portions increases the size of tables needed to manage the allocation information
  3. Having fixed-size portions simplifies the reallocation of space
  4. Having variable-size or small fixed-size portions minimizes waste of unused storage due to overallocation

Alternatives

- Two major alternatives:

Variable, large contiguous portions

- Provides better performance
- The variable size avoids waste
- The file allocation tables are small

Blocks

- Small fixed portions provide greater flexibility
- They may require large tables or complex structures for their allocation
- Contiguity has been abandoned as a primary goal
- Blocks are allocated as needed

Free Space Management

- Just as allocated space must be managed, so must the unallocated space
- To perform file allocation, it is necessary to know which blocks are available
- A disk allocation table is needed in addition to a file allocation table

Bit Tables

- This method uses a vector containing one bit for each block on the disk
- Each entry of a 0 corresponds to a free block, and each 1 corresponds to a block in use

Advantages:

- Works well with any file allocation method
- It is as small as possible

Chained Free Portions

- The free portions may be chained together by using a pointer and length value in each free portion
- Negligible space overhead because there is no need for a disk allocation table
- Suited to all file allocation methods

Disadvantages:

- Leads to fragmentation
- Every time you allocate a block you need to read the block first to recover the pointer to the new first free block before writing data to that block

Indexing

- Treats free space as a file and uses an index table as it would for file allocation
- For efficiency, the index should be on the basis of variable-size portions rather than blocks
- This approach provides efficient support for all of the file allocation methods

Free Block List

Each block is assigned a number sequentially

- The list of the numbers of all free blocks is maintained in a reserved portion of the disk

Depending on the size of the disk, either 24 or 32 bits will be needed to store a single block number

- The size of the free block list is 24 or 32 times the size of the corresponding bit table and must be stored on disk

There are two effective techniques for storing a small part of the free block list in main memory:

- The list can be treated as a push-down stack with the first few thousand elements of the stack kept in main memory
- The list can be treated as a FIFO queue, with a few thousand entries from both the head and the tail of the queue in main memory

Volumes

- A collection of addressable sectors in secondary memory that an OS or application can use for data storage
- The sectors in a volume need not be consecutive on a physical storage device
    - They need only appear that way to the OS or application
- A volume may be the result of assembling and merging smaller volumes

UNIX File Management

- In the UNIX file system, six types of files are distinguished:

Regular, or ordinary
- Contains arbitrary data in zero or more data blocks

Directory
- Contains a list of file names plus pointers to associated inodes (index nodes)

Special
- Contains no data but provides a mechanism to map physical devices to file names

Named pipes
- An interprocess communications facility

Links
- An alternative file name for an existing file

Symbolic links
- A data file that contains the name of the file to which it is linked

Inodes

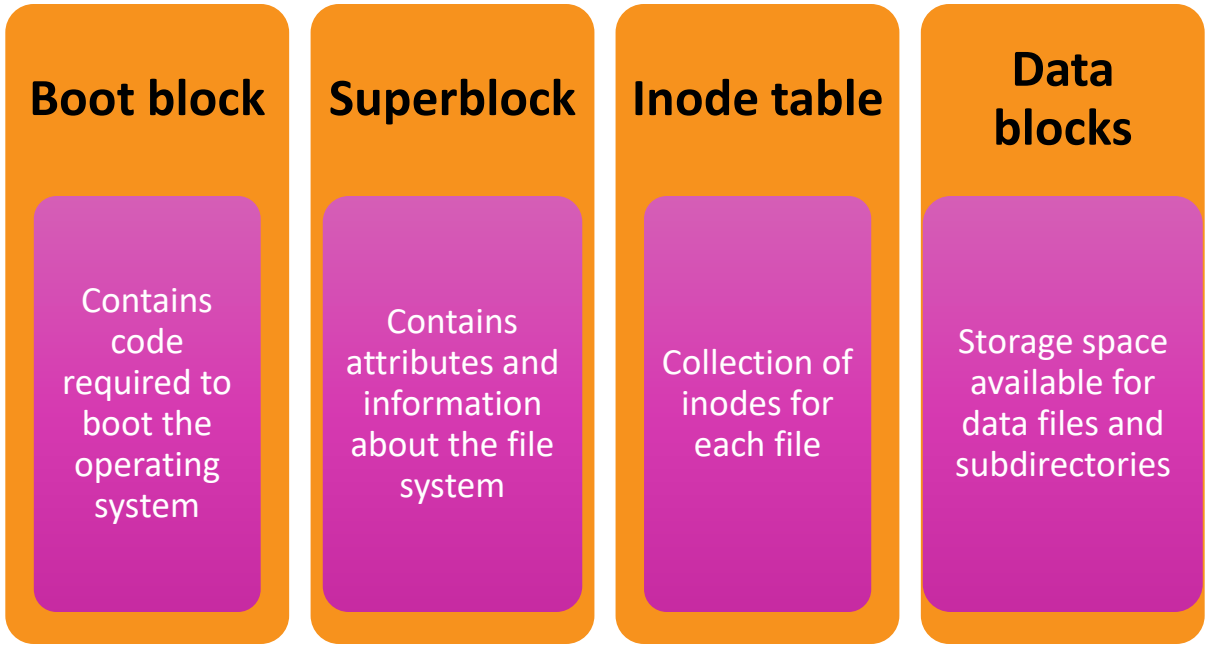
- All types of UNIX files are administered by the OS by means of inodes
- An inode (index node) is a control structure that contains the key information needed by the operating system for a particular file
- Several file names may be associated with a single inode
  - An active inode is associated with exactly one file
  - Each file is controlled by exactly one inode

File Allocation

- File allocation is done on a block basis
- Allocation is dynamic, as needed, rather than using preallocation
- An indexed method is used to keep track of each file, with part of the index stored in the inode for the file
- In all UNIX implementations the inode includes a number of direct pointers and three indirect pointers (single, double, triple)

Volume Structure

- A UNIX file system resides on a single logical disk or disk partition and is laid out with the following elements:



Windows File System

- The developers of Windows NT designed a new file system, the New Technology File System (NTFS) which is intended to meet high-end requirements for workstations and servers
- Key features of NTFS:
  - Recoverability
  - Security
  - Large disks and large files
  - Multiple data streams
  - Journaling
  - Compression and encryption
  - Hard and symbolic links

NTFS Volume and File Structure

- NTFS makes use of the following disk storage concepts:

Sector	• The smallest physical storage unit on the disk • The data size in bytes is a power of 2 and is almost always 512 bytes
Cluster	• One or more contiguous sectors • The cluster size in sectors is a power of 2
Volume	• A logical partition on a disk, consisting of one or more clusters and used by a file system to allocate space • Can be all or a portion of a single disk or it can extend across multiple disks • The maximum volume size for NTFS is 2^{64} clusters

Master File Table (MFT)

- The heart of the Windows file system is the MFT
- The MFT is organized as a table of 1,024-byte rows, called records
- Each row describes a file on this volume, including the MFT itself, which is treated as a file
- Each record in the MFT consists of a set of attributes that serve to define the file (or folder) characteristics and the file contents

SQLite

- Most widely deployed SQL database engine in the world
- Based on the Structured Query Language (SQL)
- Designed to provide a streamlined SQL-based database management system suitable for embedded systems and other limited memory systems
- The full SQLite library can be implemented in under 400 KB
- In contrast to other database management systems, SQLite is not a separate process that is accessed from the client application
  - The SQLite library is linked in, and thus becomes an integral part of the application program

Summary

- I/O devices
- Organization of the I/O function
  - The evolution of the I/O function
  - Direct memory access
- Operating system design issues
  - Design objectives
  - Logical structure of the I/O function
- I/O Buffering
  - Single/double/circular buffer
  - The utility of buffering
- Disk scheduling
  - Disk performance parameters
  - Disk scheduling policies
- Raid
- Disk cache
- UNIX SVR4 I/O
- Linux I/O
- Windows I/O

- File structure
- File management systems
- B-Trees
- File directories
  - Contents
  - Structure
  - Naming
- File sharing
- Access rights
- Simultaneous access
- UNIX file management
- Linux virtual file system
- Windows file system
- SQLite

Thank you for your attention!