UNIT 5 - INPUT/OUTPUT AND FILE MANAGEMENT

I/O Devices

Essential components in computer that facilitates communication between a user and a computer systems.

Input Devices - keyboard, mouse, scanners, microphones, digital cameras.

Output Devices - monitors, printers, speakers, projectors.

Differences between I/O Devices :-

↳ Data transfer rate ↳ Control Complexity ↳ Data representation
↳ Device application ↳ Data transfer unit ↳ Errors.

Organization of I/O Functions

Crucial for facilitating communication between CPU and peripheral devices. This organization involves several components & processes that ensure efficient data transfer, control and error management.

Mainly there are 3 modes of data transfer:

① Programmed I/O

Definition - CPU controls all data transfer directly.
How it works - CPU checks device status in a loop until its ready.
Adv - simple to implement
Disadv - Insufficient; CPU is busy waiting for I/O operations to complete.

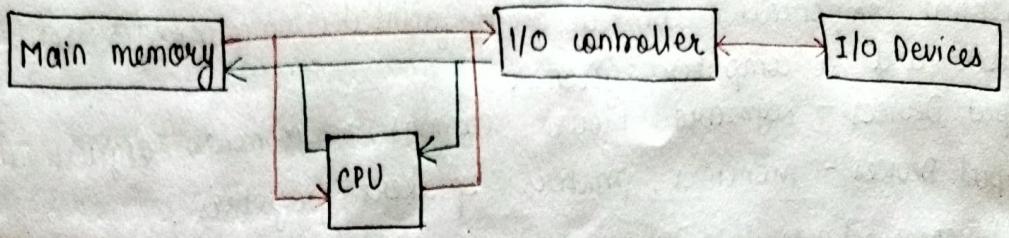
② Interrupt-Driven I/O

Definition - improves efficiency by allowing CPU to perform other tasks while waiting for I/O operation to complete. When an I/O device is ready to exchange data, it sends an interrupt signal to the CPU, which temporarily halts its current task, processes the I/O request and then resumes its previous activity.
Adv - Reduce CPU idle time.
More efficient than programmed I/O.
Disdv - Overhead from managing interrupts can introduce delays.

③ Direct Memory Access (DMA)

Definition - DMA controller manages data transfer between I/O devices & memory.
How it works - CPU initiates the transfer, then continues processing while DMA handles the data movement.
In this case CPU cannot be interrupted until complete data is transferred
Adv - High efficiency; reduce CPU involvement in data transfers.
Faster for large data blocks.

Disadv - More complex hardware design.
Potential bus contention issues.



I/O Buffering

I/O buffering involves temporarily storing data in buffer (a reserved area of memory) during transfer between CPU & I/O devices.
This process allows for smoother communication & better management of data flow.
Basically, enhances efficiency of data transfer between CPU & I/O devices.

Why I/O Buffering is needed?

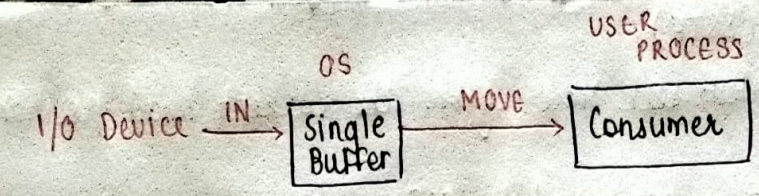
① Speed Mismatch - Buffers help to manage the difference in speed between data producers (eg modems) and consumers (eg disks).
② Data Handling - Allows for smooth data flow without interruptions, preventing data loss / delays.
③ Efficiency - Minimizes the no. of I/O operations needed by grouping data transfers.

Process of I/O Buffering.

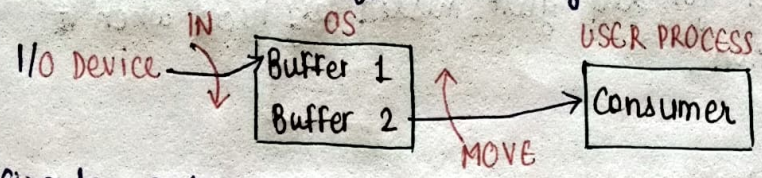
A buffer is allocated in main memory to temporarily store bytes received from modem → this incoming data is stored in buffer until it is ready to be transferred → Data from buffer to disk in single operation → while first buffer is being transferred, second buffer is used to store additional incoming data → Once first buffer is filled & its data is being transferred, the modem fills the second buffer with new data → this continues.

Approaches / Techniques in I/O Buffering

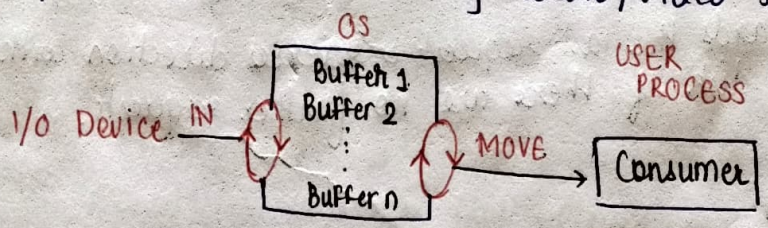
① Single Buffering
↳ Using one buffer to store data temporarily.
↳ When a process req I/O, data is read into buffer. Once the buffer is full or a specified time passes, the data is transferred to users process.
↳ Use case - suitable for simple I/O operations where data can be processed in chunks.



② Double Buffering
↳ Utilizes 2 buffers to allow overlapping of I/O operations
↳ When 1 buffer is being filled with incoming data, the other buffer is being emptied (processed). This minimizes idle time & improves efficiency.
↳ Use Case: video streaming / real time data processing
↳ It is also called buffer swapping.



③ Circular Buffering
↳ Multiple buffers arranged in circular manner.
↳ When one buffer is filled, it wraps around to overwrite older data, allowing continuous inputs and outputs without waiting for processing to complete.
↳ Use case: For handling bursty I/O operations where data arrives at unpredictable intervals, eg audio/video streaming., communication protocols



fifo manner

Disk Scheduling

Manages the order of disk I/O requests to optimize performance & minimize access time.

Disk Scheduling Algorithms.

① First Come first Served (FCFS)
↳ Requests are processed in order they arrive
↳ Adv - Simple & easy to implement.
No starvation; every request is eventually serviced.
↳ Disadv - Can lead to long wait times (convoy effect).
Inefficient for large number of requests.

② Shortest Seek Time first (SSTF)

↳ The request with the shortest seek time from the current head position is served first.

↳ Adv - Reduce total seek time compared to FCFS.

↳ Disadv - Starvation may occur.

③ SCAN (Elevator Algorithm)

↳ The disk arm moves in 1D servicing requests until it reaches the end of the disk and then reverses direction.

↳ Adv - Efficient than FCFS; reduce wait times for req near ends.

↳ Disadv - long wait times for req just missed by head when it reverses direction.

④ C-SCAN (Circular Scan)

↳ Similar to scan, but after reaching one end, it jumps back to other end without servicing requests on return trip.

↳ Adv - Provides uniform waiting time.
Better response time compared to SCAN.

↳ Disadv - More seek movements
Can lead to longer wait times for req. at far ends

⑤ LOOK

↳ The disk arm moves towards the last request & reverses direction without going to the end of the disk if there are no further requests.

↳ Adv - More efficient
Avoids starvation

↳ Disadv - Overhead in locating the last request.
Requests far from the current position may experience long wait times.

⑥ C-LOOK

↳ Similar to Look, but when servicing is complete, it jumps back to last request instead of moving all the way to the end of the disk.

↳ Adv - Efficient with less waiting time.
Avoids unnecessary movement

↳ Disadv - Still has overhead in finding end requests.

FILE AND FILE SYSTEMS

↳ digital container used to store information on a computer / other electronic devices.

files contain various data types including docs, images, video, etc.

Each file is identified by unique name & often has an associated file extension that indicates its format (eg. '.txt' '.jpg' '.ipynb', etc)

Importance of files

↳ Store info persistently.
↳ Retrieve & manipulate data easily.
↳ Share info across different systems & applications.

File System – is a method & structure used by an os to manage files on storage device like HDD, SSD, USB. It provides a way to organize files into directories & manage how data is stored & retrieved.

functions of File Management Systems

↳ File creation / deletion
↳ file organization
↳ file access control
↳ file sharing
↳ Data security
↳ file Retrieval
↳ file Backup & Retrival
↳ File Compression
↳ Space management
↳ File Metadata management
↳ Auditing & Logging.

File Structure

ie how is data organized, stored & accessed within a file in computer sys.

Components :—

↳ Data Representation
↳ file format (eg. CSV)
↳ Access Method
↳ file operations
↳ Metadata Management

Types of file structures :—

① Flat file.s.

A simple structure where each record is stored as single line in a text file.

Eg. A log file where each entry is a newline

② Hierarchical file.s.

Organizes files in a tree-like structure with directories (folders) containing subdirectories & files.

Eg. files that are organised into folders.

③ Relational file.s.

Data is organised into tables with relationships betn them, commonly used in db.

Eg. A database system where user info in 1 table & orders info in 2 table.

④ Indexed file·s·
Uses an index to allow quick access to records within file based on key values
Eg· A library catalog.

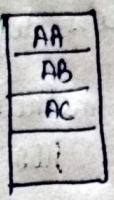
⑤ Multilevel Indexing
Multiple levels of indexing

File Organization & Access → Sequential Access
→ Random
→ Other.

How is data organized in a file system & how to access it.

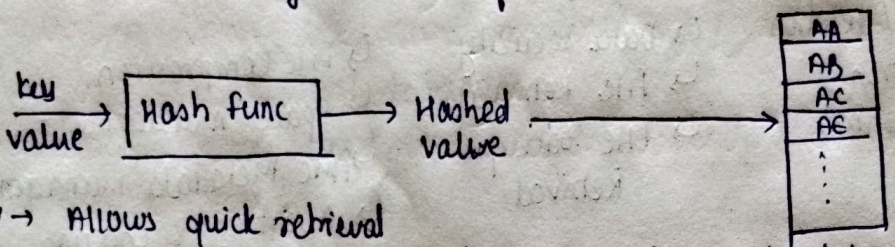
① Sequential file organization.
Info is stored one after another

Adv → simple design, easy sorted



Disadv → time consume → overhead

② Direct file organization
Records are stored randomly but accessed directly using a unique key that determines the storage location of each record.



Adv → Allows quick retrieval
Support large variable sized records, enhance flexibility.

Disadv → Complexity in managing storage locations.
May lead to fragmentation.

③ Indexed file organization
Index is maintained that maps keys to their corresponding record location, allowing faster access

Adv → less search time
supports both sequential & random access methods

Disadv → Additional storage space required
Updating is slow

④ Heap file organization
Records are inserted at the end of the file without any specific order / sorting.

Adv → Simple & efficient
Insertion is fast

Disadv → Ineffective for searching
Waste space.

⑤ Indexed Sequential Access Method.

File Directories

Files are often grouped into directories (or folders) that provides a way to organize files and manage file hierarchy.

It serves as a catalog that contains references to other files on a storage device.

Each entry in a directory includes data about files ie names, types, sizes location & permission.

function :-

- ↳ organization
- ↳ Metadata management
- ↳ file operations (update, delete, rename, etc)
- ↳ Access Control (Manage permission that who can modify certain files)

Operations performed on directory :-

- ↳ Create
- ↳ Delete
- ↳ Rename
- ↳ Listing directory contents
- ↳ Searching for a file
- ↳ Linking & Unlinking files
- ↳ copy
- ↳ Move
- ↳ compress

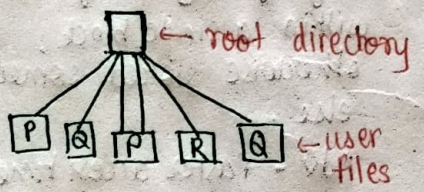
Directories are of 3 types :-

① Single-Level Directory

All files are in 1 directory

Adv - simple & easy to navigate
- Reduce Redundancy

Disadv - cannot have files with same name
- hard to manage if files increase

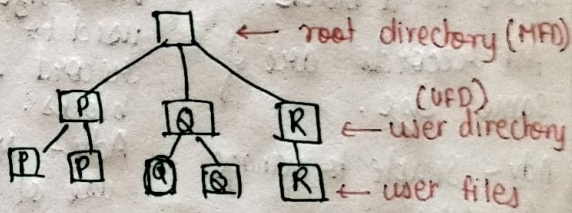


② Two-level Dir.

Each user has its UFD (user file dir) under a MFD (Master file dir)

Adv - Allows multiple users to have same file name
- More secure

Disadv - No grouping ability
- No 2 files for 1 user can have same name

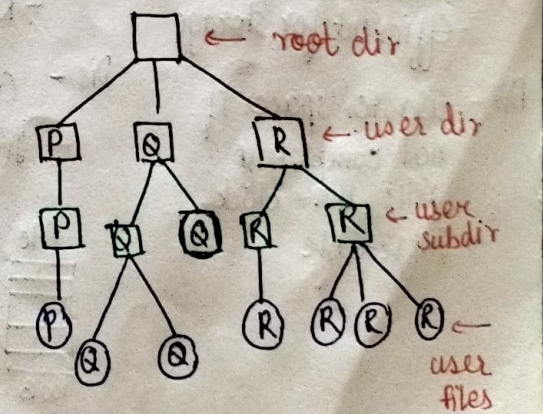


③ Tree-Level / Structured Directory.

files are organized in a hierarchical tree structure where dir. can contain subdir.

Adv - Support logical grouping of related files.
- Allow multiple files with same name.

Disadv - Complexity increases with deeper hierarchies.



File Sharing

The practice of distributing / providing access to files across various platforms / networks.

File Sharing permissions :-

① Read permission
can User can view but not modify the content.
Eg. reports, ppt, reference materials

② Write Permission
User can modify.
Eg. documents

③ Execute Permission
Users can run executable file / scripts.
Eg. command-line scripts.

Applications of file Sharing :-

↳ Collaborative platforms - eg. Google Drive, Collab.
↳ Network file sharing - eg. File Zilla, Cyberdunk
↳ Email Attachments.

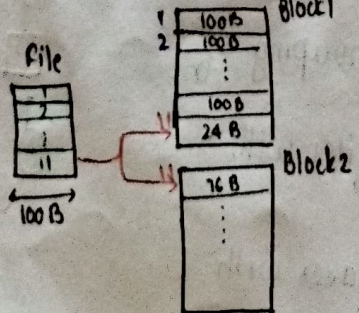
Record Blocking

↳ organizing / storing file / record in fixed-size / variable-size blocks within a file system / database.
↳ Each block contain one / more records, depending on size of the record & the block.

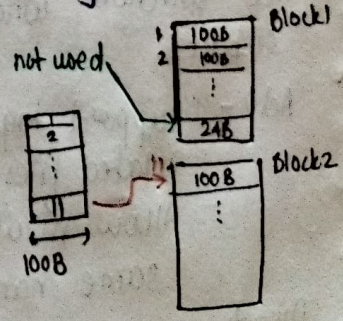
Types of RB :-

① fixed length
→ Each record occupies fixed amount of space in a block.
↳ All records are of same size.
Adv - Simple to manage & access
- Efficient for storage
Dis - Wastes spaces if not completely fills the block.
- Inflexible

② Variable length Spanned
Records can span multiple blocks if they exceed the block size. A pointer is used to link parts of a record stored in diff. blocks.
Adv - More efficient use of space.
- Reduce internal fragmentation
Dis - Complex logic
- Increase overhead



③ Variable length Unspanned
Records are stored in blocks only if they can fit entirely within a single block; otherwise stored somewhere else.
Adv - Faster access time
- Reduce complexity
Dis - Leads to internal fragmentation



Secondary Storage Management

SSM includes devices like HD, SSD, optical disks

File Allocation methods

① Contiguous Allocation

files are stored in continuous blocks.

A - fast access, simple management.

D - External fragmentation.

② Linked Allocation

files are linked list of blocks.

A - No external fragmentation.

D - Slow access, only sequential access.

③ Indexed Allocation

An index block contains pointers to file blocks.

A - Efficient random access, no ext. fragmentation.

D - Overhead for large files, complex for large files.

④ file Allocation Table

A table tracks which blocks belong to which files.

A - Efficient random access.

D - Overhead for large files.

⑤ Inode-Based Allocation

Metadata & pointers stored in inodes.

A - Detailed management & efficient access.

D - fixed no. of inodes limits file count.

Free space Management

① Bit Map / Bit Vector

Uses bits to represent free (0) & occupied (1) blocks

A - Simple & efficient

D - Required extra space for bitmap.

② Linked List

Free blocks are linked together with pointers

A - Efficient space usage

D - Slower access time due to traversal

③ Free Space List

A list of all free blocks

A - Simple management

D - Can become insufficient with scattered free space.

④ Grouping

Groups multiple free blocks together with pointer

A - Quick access

D - Requires some traversal.