*Deadlocks are situations in which a set of processes is stuck waiting for resources that each hold, creating a cycle of dependencies that prevents any process from progressing. Here are the primary methods to handle deadlocks in operating systems:*

***1. Deadlock Prevention***

***Aim:*** *Prevent any of the necessary conditions for deadlock from occurring.*

***(Separate Question can be made from this part)***

*Deadlocks occur only if the following four conditions are met simultaneously:*

***1. Mutual Exclusion:*** *Only one process can use a resource at a time.*

***2. Hold and Wait:*** *Processes holding resources can request additional resources.*

***3. No Pre-emption:*** *Resources cannot be forcibly taken from processes.*

***4. Circular Wait:*** *A closed chain of processes exists, where each process holds at least one resource that the next process in the chain needs.*

*By preventing one or more of these conditions, we prevent deadlocks:*

***Mutual Exclusion:*** *Share resources where possible.*

***Hold and Wait:*** *Require processes to request all resources at once.*

***No Pre-emption:*** *Allow resources to be pre-empted if necessary.*

***Circular Wait:*** *Impose a strict order on resource acquisition.*

***2. Deadlock Avoidance***

***Aim:*** *Allow processes to request resources dynamically but ensure that the system remains in a safe state.*

***Banker’s Algorithm:***

*This is a popular deadlock avoidance method where a process declares the maximum number of resources it will need.*

*The system only allocates resources if doing so will leave the system in a safe state (a state where all processes can complete without deadlock).*

***Safe State:*** *A state in which the system can allocate resources to each process up to its maximum, allowing each process to finish without causing deadlock.*

***Drawbacks:***

*Works best when the maximum resource needs of each process are known in advance.*

*Can be complex and require significant overhead to check for safe states frequently.*

***3. Deadlock Detection and Recovery***

***Aim****: Detect deadlocks when they occur and then recover from them.*

***Detection Algorithms:***

*For example, a wait for graph is used to represent resource allocation and wait dependencies.*

*The system periodically checks for cycles in the wait for graph. If a cycle is found, a deadlock exists.*

***Recovery Methods:***

***Process Termination:*** *Terminate one or more processes involved in the deadlock. This can be done in two ways:*

***Abort All Deadlocked Processes:*** *Drastic but guarantees removal of deadlock.*

***Abort Processes One at a Time:*** *Abort processes one by one, checking after each to see if the deadlock is resolved.*

***Resource Pre-emption:*** *Take resources from one process and give them to others until the deadlock is resolved. Considerations include:*

***Rollback:*** *Roll back a process to an earlier safe state if resources are pre-empted.*

***Priority:*** *Consider process priority or costs associated with rollback when deciding which process to pre-empt.*

***4. Deadlock Ignorance***

*In this approach, the system simply assumes that deadlocks are rare enough to be ignored.*

***Used by Operating Systems like Unix and Windows:*** *Deadlock detection and recovery are not often implemented in general-purpose operating systems, as the cost of these algorithms can be high.*

*When a deadlock occurs, it might lead to a process hang, requiring manual intervention (e.g., terminating processes).*

*Each method has trade-offs and is chosen based on factors such as the system’s purpose, the resources it manages, and performance requirements.* ***For example,*** *Deadlock Prevention and Avoidance are generally preferred in critical systems where uptime and resource management are essential. In contrast, Deadlock Detection and Recovery or Ignorance may be acceptable in less critical or user interactive systems.*

***What is Authentication?***

*The process of verifying whether a person is a legitimate user or not. It ensures only authorized users access the system.*

***Ways to Authenticate:***

***Something you know:*** *A secret known only to the user, like passwords or PINs.*

***Something you have:*** *A physical object, such as a magnetic card, RFID badge, or token.*

***Something you are:*** *Unique characteristics of the user, also known as biometrics.*

***Types of Biometrics:***

***Static Biometrics:*** *Based on the user's physical traits. Examples:*

*Fingerprints*

*Facial features*

***Dynamic Biometrics:*** *Based on the user's behavior or actions. Examples:*

*Handwriting patterns*

*Voice recognition*

*Typing rhythm*

***File Access Control***

***Purpose of File Access Control:***

* + *Operating systems protect file data since it is a primary concern in computer security.*
  + *Determines who can access files, what actions they can perform, and under what conditions.*

***Access Rights for Files and Devices:***

* 1. ***For Files:***

***AOS/VS System (OWARE):*** *Own, Write, Append, Read, Execute.*

***UNIX System (RWX):*** *Read (R), Write (W), Execute (X).*

* 1. ***For Devices (e.g., Printers):***

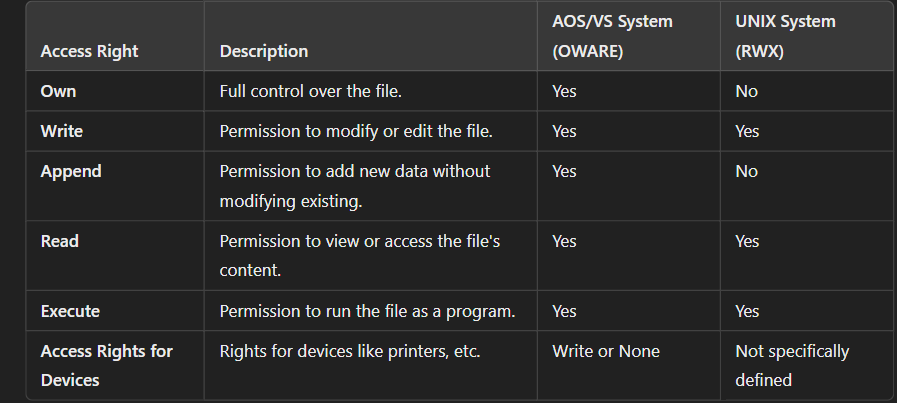
*Only "Write" or "None" access rights are applicable.*

***List of Access Rights:***

* + ***Read:*** *Permission to view the file's content.*
  + ***Write:*** *Permission to modify the file.*
  + ***Execute:*** *Permission to run the file as a program.*
  + ***Append:*** *Permission to add new data to the file without modifying existing data.*
  + ***Own:*** *Full control over the file.*

*A screenshot of a computer

Description automatically generated*

**

***Cryptography***

***What is Cryptography?***

* + *The art of converting plain text into unreadable text (encryption) and then back into readable text (decryption) to ensure secure communication.*
  + *Protects data from unauthorized access during storage and transmission.*

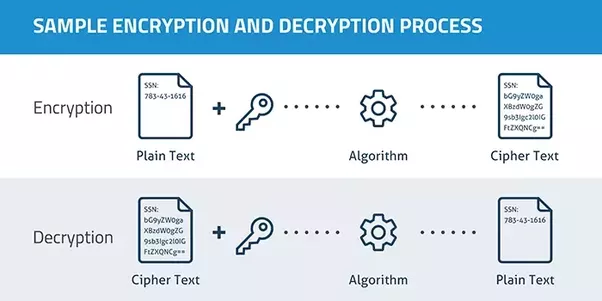
***Key Components of Cryptography:***

* + ***Encryption:*** *Converts original data (plaintext) into unreadable data (ciphertext).*
  + ***Decryption:*** *Converts ciphertext back into plaintext.*
  + ***Key:*** *A secret code used in both encryption and decryption. It is a sequence of random bits.*

***Terms to Remember:***

***Plaintext:*** *Original, readable data before encryption.*

***Ciphertext:*** *Encrypted, unreadable data after encryption.*



***Private Key Cryptography (Symmetric Cryptography):***

***Definition:***

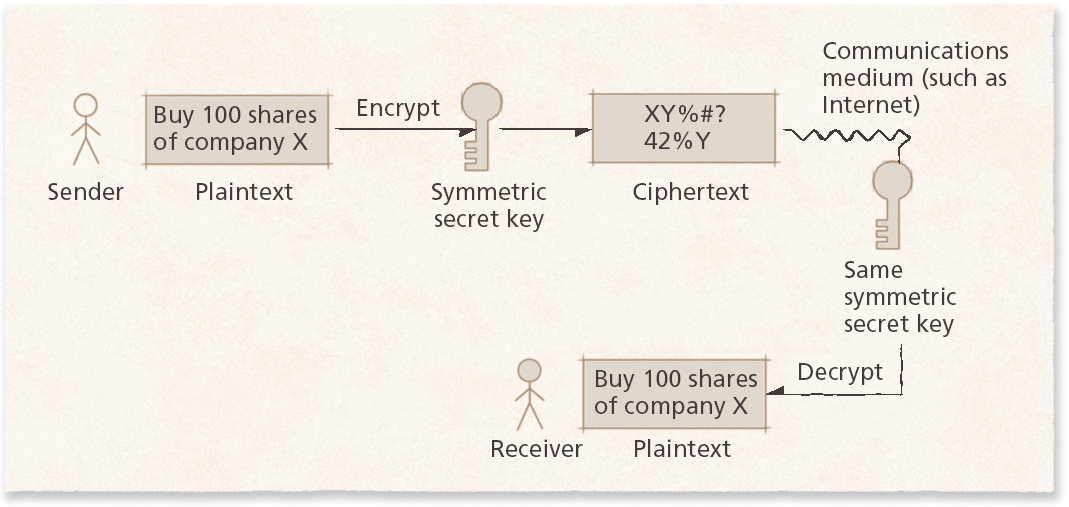
*A method where the same secret key is used for both encryption and decryption.*

***How it Works:***

* + *The sender encrypts the message using the secret key.*
  + *The receiver decrypts the message using the same key.*

***Limitations:***

* + *The key must be shared securely between parties before communication can happen.*
  + *If the key is intercepted during sharing, the system becomes insecure.*



***Public Key Cryptography (Asymmetric Cryptography):***

***Definition:***

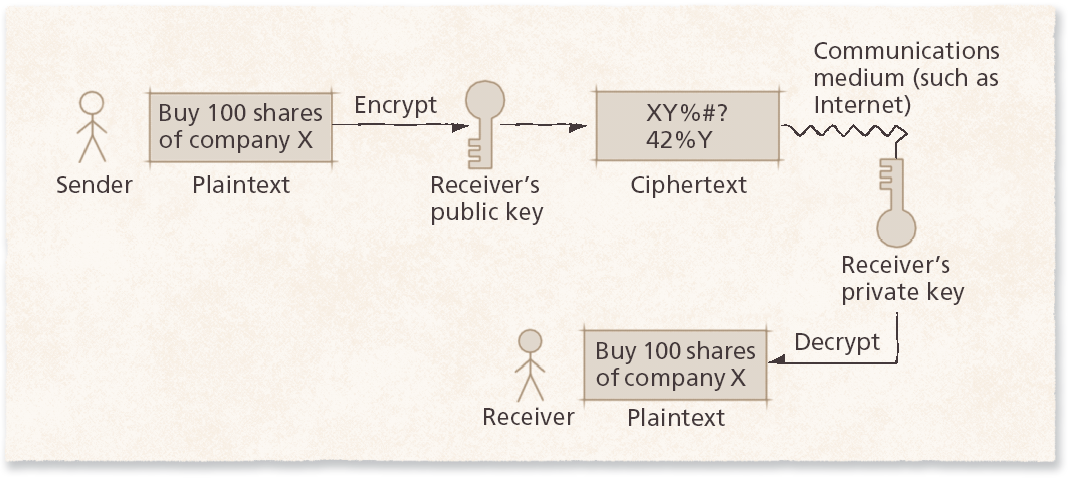
*A cryptographic method that uses two keys: a* ***public key*** *and a* ***private key.***

***How it Works:***

* *The public key is shared openly and used for encryption.*
* *The private key is kept secret and used for decryption.*
* ***Example Workflow:***
  + 1. *Sender uses the receiver's public key to encrypt a message.*
    2. *Receiver uses their private key to decrypt the message.*

***Advantages:***

* + *Eliminates the need to securely exchange a shared secret key.*
  + *Secure even if the public key is widely distributed.*



***1. Introduction to Memory Management:***

* ***Definition:*** *Management of primary memory (main memory/real memory) in a computer system is crucial for operating system (OS) design.*
* ***The Process of controlling and coordinating computer memory, assigning portions known as blocks to various running programs to optimize the overall performance of system is called memory management.***
* ***Purpose:*** *Programs and data must be in main memory to execute or be directly referenced.*
* ***Storage Types:***

***Main Memory:*** *Fast, but expensive and limited in capacity.*

***Secondary Storage:*** *Inexpensive and large capacity, typically used for data not immediately needed (e.g., disk storage).*

***2. Storage Organization:***

* ***Main Memory:*** *Viewed as an expensive and valuable resource.*
* ***Key Questions for Design:***
  1. *Should main memory hold one user or multiple users?*
  2. *Should each user receive equal space, or should it be divided into partitions of different sizes?*
  3. *Should partitions be static or dynamically allocated?*
  4. *Should jobs require contiguous memory, or can they be split and stored in separate memory slots?*

***3. Storage Management Strategies:***

***Fetch Strategies:***

* *Determine when programs or data are loaded into memory.*

1. ***Demand Fetch:***
   * *Loads data only when referenced.*
   * ***Advantages:*** *Simple to implement.*
   * ***Disadvantages:*** *Process cannot run until the block is brought into memory.*
2. ***Anticipatory Fetching:***
   * *OS predicts which blocks are needed and loads them in advance.*
   * ***Advantages:*** *Reduces process suspension if predictions are correct.*
   * ***Disadvantages:*** *May preload unnecessary blocks due to unpredictable process behaviour.*

***Placement Strategies:***

* *Decide where to place programs in memory.*

1. ***First-Fit:*** *Place in the first suitable hole.*

* ***Advantages:*** *Low overhead, simple to implement.*

1. ***Best-Fit:*** *Place in the smallest hole that fits the process.* 
   * ***Advantages:*** *Minimizes unused space.*
   * ***Disadvantages:*** *Higher time overhead.*
2. ***Worst-Fit:*** *Place in the largest available hole.* 
   * ***Advantages:*** *Leaves large holes for future processes.*
   * ***Disadvantages:*** *Creates fragmentation.*
3. *Next-Fit*

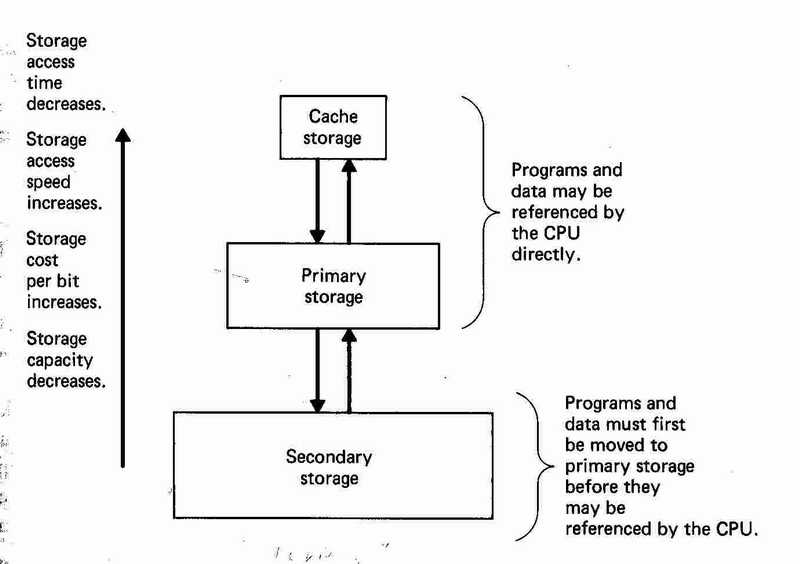
***Replacement Strategies:***

* *Decide which program/data to remove when memory is full.*

1. ***Optimal:*** *Replaces data not needed for the longest time.*
2. ***FIFO (First-In-First-Out):*** *Removes the oldest data.*
3. ***LRU (Least Recently Used):*** *Replaces data not used recently.*
4. ***Clock:*** *Circular replacement strategy using a reference bit.*

***4. Storage Hierarchy:***

* ***Hierarchy Levels:***
  1. ***Main Memory:*** *Stores currently needed program instructions and data.*
  2. ***Cache Memory:*** *High-speed, small-sized storage located on the processor, used for frequently accessed data.*
  3. ***Secondary Storage:*** *Large capacity, used for data and programs not actively needed.*
* ***Key Concept:*** *Frequently used data moves between these levels for optimal performance (e.g., caching for speed).*



***5. Contiguous vs. Non-Contiguous Allocation:***

***Contiguous Allocation:***

* ***Definition:*** *A program must occupy a single block of memory.*
* ***Characteristics:***
  + *Simpler implementation.*
  + *Prone/Suffer to fragmentation, making it harder to find large enough blocks.*
* ***Overhead:*** *Minimal.*

***Non-Contiguous Allocation:***

* ***Definition:*** *Programs are divided into chunks (segments or pages) placed in different memory locations.*
* ***Advantages:*** *Easier to find space for smaller pieces, allowing multiple programs in memory.*
* ***Disadvantages:*** *Higher overhead for tracking locations.*

***6. Fixed Partition Multiprogramming:***

***Characteristics:***

1. *Divides main memory into fixed-sized partitions.*
2. *Each partition holds one job.*
3. *Partitions are statically defined and cannot adapt to process size changes.*

***Types:***

1. ***Absolute Translation:***
   * *Programs are designed to run in specific partitions only.*
   * ***Drawback:*** *Internal fragmentation if jobs don’t fully utilize partitions.*
2. ***Relocatable Translation:***
   * *Jobs can run in any partition that fits their size.*
   * *Requires* ***base and limit registers*** *for addressing.*

***7. Variable Partition Multiprogramming:***

***Characteristics:***

* *Eliminates fixed boundaries.*
* *Allocates just enough memory for each job, reducing internal fragmentation.*
* ***Disadvantage:*** *Can lead to external fragmentation.*

***Solutions to External Fragmentation:***

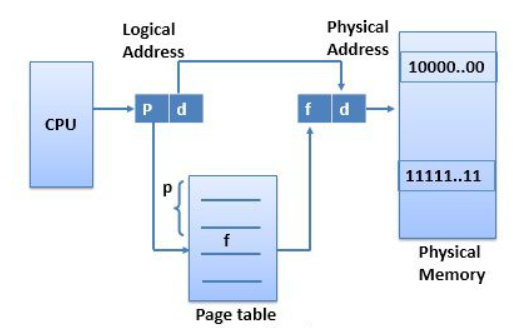
1. ***Coalescing Holes:*** *Combine adjacent free memory blocks.* 
   * ***Limitation:*** *Often insufficient for significant memory recovery.*
2. ***Compaction:*** *Rearranges memory into contiguous blocks.* 
   * ***Limitation:*** *High system resource consumption, interrupts processes.*

***8. Address Translation in Memory Management:***

* ***Logical Address:*** *Independent of the actual memory location.*
* ***Physical Address:*** *Actual location in main memory.*
* ***MMU (Memory Management Unit):*** *Translates logical addresses to physical addresses.*

***Contiguous Addressing:***

1. ***Base Register:*** *Starting address of a process.*
2. ***Limit Register:*** *Defines the end of a process’s memory space.*
3. ***Translation Process:*** *Logical address + base = physical address.*



***9. Paging:***

***Definition: Divides memory and processes into fixed-size chunks (pages/frames).***

* ***Advantages:*** *Reduces internal fragmentation compared to contiguous allocation.*
* ***Disadvantages:*** *Still incurs overhead for maintaining page tables.*

***Translation:***

* *Page number maps to a frame number using the page table.*
* *Offset remains the same.*

***10. Segmentation:***

***Definition: Divides programs into variable-sized logical units called segments.***

* ***Advantages:*** *Eliminates internal fragmentation.*
* ***Disadvantages:*** *External fragmentation occurs.*

***Translation:***

* *Uses a segment table to map segment number + offset to the physical address.*

A diagram of a computer program

Description automatically generated

*A white box with black text

Description automatically generated*

***11. Combined Paging and Segmentation:***

* ***Definition:*** *Combines the benefits of paging and segmentation.*
* ***Process:***
  + *Segments are divided into pages of fixed size.*
  + *Paging is transparent to the programmer; segmentation remains visible.*

***1. Files***

* + *Files are the central element to most applications.*
  + *The file system is one of the most important parts of the OS to a user.*

***Desirable properties of files:***

* + 1. *Long-term existence.*
    2. *Sharable between processes.*
    3. *Structure.*

***Explanation:*** *Files are essential because they store data for applications. They are designed to exist independently of the system being powered on, to be used by multiple processes, and to have an organized structure (e.g., text, binary).*

***2. File Management System***

* + *File management system consists of system utility programs that run as privileged applications.*
  + *Concerned with secondary storage.*

***Typical Operations:***

* + *Create, Delete, Open, Close, Read, Write.*
  + ***Create:*** *Make a new file.*
  + ***Delete:*** *Remove an existing file.*
  + ***Open/Close:*** *Access or finish using a file.*
  + ***Read/Write:*** *View or edit a file’s contents.*

***Explanation:*** *A file management system handles file-related tasks on secondary storage, such as hard drives. Operations like creating or reading files are critical for users and applications.*

***3. Terms related to FMS***

* 1. ***Field:*** *Basic element of data, contains a single value, characterized by its length and data type.*
  2. ***Record:*** *Collection of related fields, treated as a unit.*
  3. ***File:***
     + *Has file names.*
     + *A collection of similar records or data.*
     + *Treated as a single entity.*
     + *May implement access control mechanisms.*

***Explanation:***

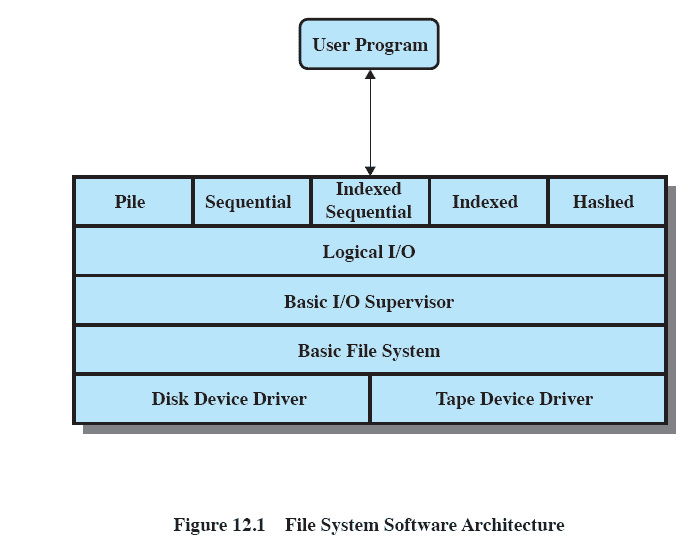
* *Fields are the smallest pieces of data (e.g., a person’s name or age).*
* *Records combine related fields into meaningful units (e.g., a person’s profile).*
* *Files store these records, often with security controls like permissions.*

***4. File System Software Architecture***

* + ***Device Drivers:*** *Lowest level communicates with peripheral devices, responsible for starting I/O operations on a device, processes I/O completion.*
  + ***Basic File System or Physical I/O:***
    - *Primary interface with the environment outside the computer system.*
    - *Deals with exchanging blocks of data.*
    - *Concerned with placement and buffering of blocks.*
  + ***Basic I/O Supervisor:***
    - *Responsible for all file I/O initiation and termination.*
    - *Deals with device I/O scheduling and file status.*
  + ***Logical I/O:***
    - *Enables users and applications to access records.*
    - *Maintains basic data about files.*
  + ***Access Methods:***
    - *Closest to the user.*
    - *Reflect different file structures.*
    - *Provides a standard interface between applications and file systems/devices.*

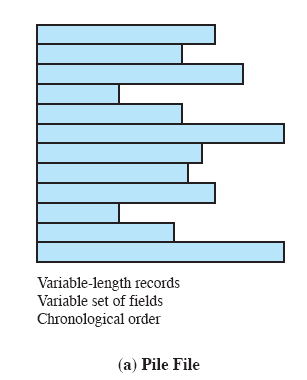
***Explanation:*** *The architecture defines layers of the file system:*

* ***Device Drivers:*** *Handle hardware communication.*
* ***Basic File System:*** *Manages data blocks and placement.*
* ***Basic I/O Supervisor:*** *Manages scheduling and status tracking.*
* ***Logical I/O:*** *Provides access to file details and records.*
* ***Access Methods:*** *Define how applications interact with files.*



***5. File Organization***

* + *Refers to the logical structure of records.*
  + *Determined the way in which files are accessed.*
  + ***Criteria for File Organization:***
    1. *Short access time.*
    2. *Ease of update.*
    3. *Economy of storage.*
    4. *Simple maintenance.*
    5. *Reliability.*
* ***Types of File Organization:***
* ***Pile Files:***
  + *Data is stored in the same order it is received, without any specific structure.*
  + *The purpose is to gather and save a large amount of raw data.*
  + *Records in the file can have different fields (not standardized).*
  + *To find a record, the system must search through the entire file, which is slow (called "exhaustive search").*
  + *These files are often used to temporarily store data before it is processed.*
  + ***Limitation:*** *They are not suitable for most applications because searching through them is inefficient.*

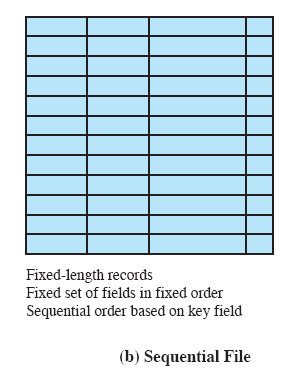


***Sequential Files***

*These files are organized in a specific order, making them easy to understand and process for certain tasks.*

1. ***Structure****:*
   * *All records (data entries) in the file have the* ***same size*** *and contain the* ***same type of information*** *(e.g., name, ID, or address).*
   * *Each record has a* ***key field****, like an ID number or name, which makes it unique.*
   * *The records are stored in a* ***specific order*** *based on the key (e.g., alphabetically by name or numerically by ID).*
2. ***How They’re Used****:*
   * ***Best for batch tasks*** *where you process all records at once, like sending bills to customers or calculating employee salaries.*
   * ***Not good for quick tasks****, like searching for or updating a single record, because you have to scan the file sequentially (one record at a time).*
3. ***Adding New Records****:*
   * *New records (e.g., new employees or customers) are stored temporarily in a separate file called a* ***log file****.*
   * *Later, the log file is merged with the main file to maintain the order of records.*

*This makes sequential files great for organized tasks but slow for searching or updating specific data.*



***Indexed Sequential File***

*This type of file combines the benefits of* ***sequential files*** *(records are stored in order) with some extra features to make searching faster.*

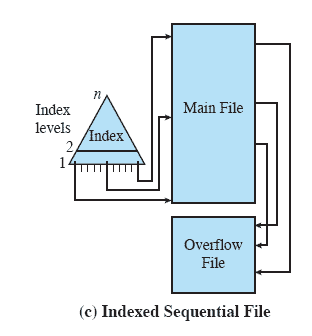
1. ***What Stays the Same****:*
   * *Just like sequential files, the records are organized in a specific* ***order based on a key field*** *(e.g., ID, name).*
2. ***New Features Added****:*
   * ***Index****:*
     + *Think of the index like a table of contents in a book. It helps you* ***quickly find the area where the record is located****.*
     + *Example: Instead of reading every page of a book to find a chapter, you check the table of contents to jump directly to the right section.*
   * ***Overflow File****:*
     + *If there’s no space to add a new record in the main file, the record is stored in an* ***overflow file*** *(a separate area).*
     + *The main file contains a* ***pointer*** *that links to the record in the overflow file, so it’s still connected.*

***How It Works:***

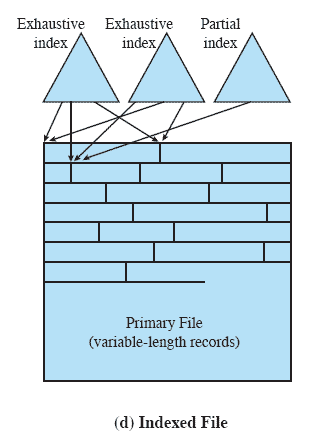
1. ***Index****:*
   * *The index itself is a small file.*
   * *Each entry in the index has two parts:* 
     + ***Key field****: The key (e.g., ID number) to identify the record.*
     + ***Pointer****: This points to the location of the record in the main file.*
2. ***Searching for a Record****:*
   * *Instead of searching the entire file, the system looks at the* ***index*** *to find the key field.*
   * *The* ***pointer*** *in the index tells the system where to find the actual record in the main file.*
3. ***Main File****:*
   * *Just like the sequential file, records are stored in order in the main file.*
   * *Each record in the main file has a hidden field called a* ***pointer****, which links to the overflow file in case the record is stored there.*

***Benefits:***

* ***Faster Access****: The index makes searching for individual records much quicker compared to sequential files.*
* ***Organized Structure****: The file still keeps its sequential order, which is useful for batch processing.*



* ***Indexed File:***
  + ***Uses multiple indexes*** *for different key fields (e.g., one index for names, another for IDs).*
  + *May contain:*
    1. *An* ***exhaustive index*** *(one entry for every record in the main file).*
    2. *A* ***partial index*** *(only contains entries for a subset of records).*
  + *When a* ***new record*** *is added to the main file,* ***all index files must be updated*** *to maintain consistency.*
  + ***Access:*** *Records can only be accessed through their* ***indexes****.*
  + ***No restrictions*** *on where records can be placed in the main file, if there is a pointer in at least one index referring to that record.*
  + ***Supports variable-length records*** *(records don’t have to be fixed in size).*



* + ***Direct/Hashed File:***
  + ***Exploits disk capabilities*** *to directly access any block of memory at a known address.*
  + *Uses* ***hashing*** *on the key value to determine the location of records.*
  + ***Access:*** *Directly access a block at a known address using a* ***mathematical formula*** *(hash function).*
  + ***Key field*** *is required for each record to apply the hashing method.*

***6. File Directory***

* *A* ***file directory*** *is a collection of files associated with a file management system.*
* *The directory stores information about the files, such as:*
  + ***Attributes*** *(e.g., file type),*
  + ***Locations*** *(where the file is stored),*
  + ***Ownership*** *(who owns the file).*
* *Much of this information, especially related to storage, is managed by the* ***Operating System (O/S)****.*
* *The* ***directory itself*** *is a file owned by the O/S and is accessible by various file management routines.*
* ***From the user’s perspective,*** *the directory provides a way to* ***map file names*** *(known to users and applications) to the actual files.*
* *Typically, the* ***file owner*** *can grant access to other users.*

***Elements of a File Directory:***

* + ***Basic Information:*** *File name, type, organization.*
  + ***Address Information:*** *Device, starting address, size.*
  + ***Access Control Information:*** *Owner, permitted actions, and user permissions.*
  + ***Usage Information:*** *Dates (creation, last access), activity logs, etc.*

***Basic Information***

* *File Name* 
  + *Name as chosen by creator be unique with in a directory*
* *File Type* 
  + *Text file, Binary file*
* *File Organization* 
  + *For systems that support different file organizations.*

***Address Information***

* *Volume* 
  + *Indicate device on which file is stored.*
* *Starting Address* 
  + *Starting physical address on secondary storage for example cylinder track and block number on disk.*
* *Size Used* 
  + *Current size of the file in bytes, words or blocks.*
* *Size Allocated*
  + *The maximum size of the file.*

***Access Control Information***

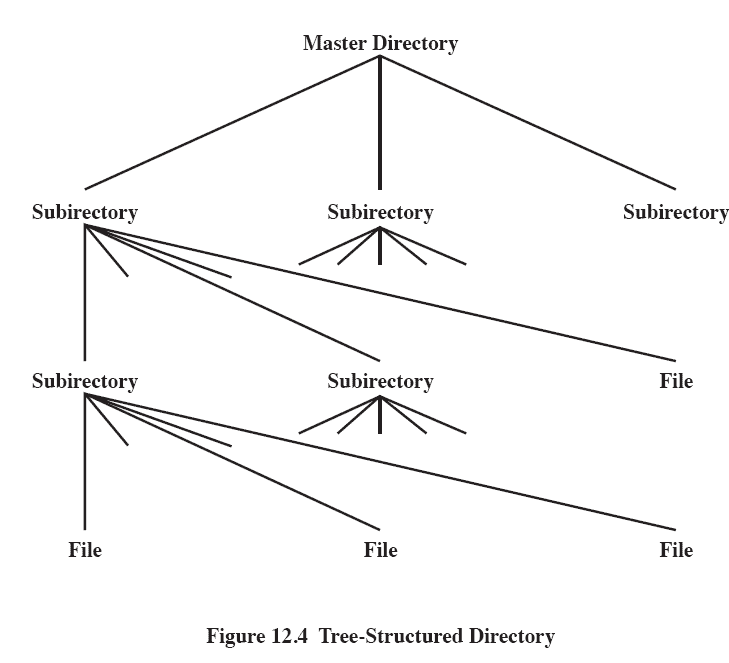
* *Owner* 
  + *The owner may be able to grant/deny access to other users and to change these privileges.*
* *Access Information*
  + *May include the user’s name and password for each authorized user.*
* *Permitted Actions* 
  + *Controls reading, writing, executing, transmitting over a network*

***Usage Information***

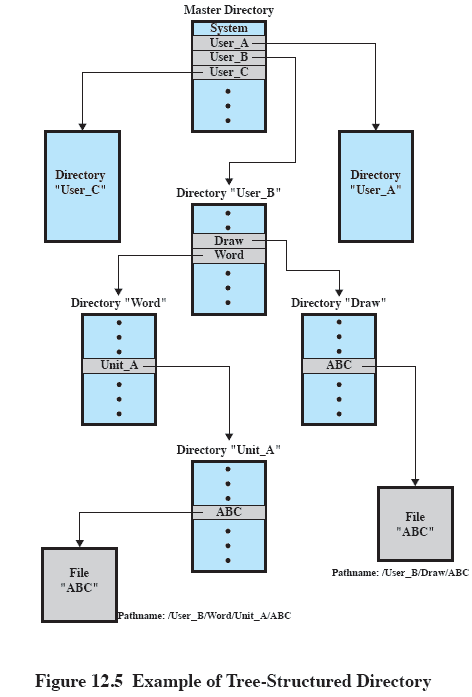
* *Date Created*
* *Identity of Creator*
* *Date Last Read Access*
* *Identity of Last Reader*
* *Date Last Modified*
* *Identity of Last Modifier*
* *Date of Last Backup*
* *Current Usage* 
  + *Current activity, locks, etc*

***Structure for a Directory***

* *The method for storing the previous information varies widely between systems*
* *Universally adopted approach is to use a Hierarchical or Tree Structure*
* *Master directory with user directories underneath it*
* *Each user directory may have subdirectories and files as entries*



***Example of   
Tree-Structured Directory***



***Operations***

***1. Operations Performed on a Directory:***

*A directory system should support the following operations:*

* ***Search:*** *Find specific files in the directory.*
* ***Create Files:*** *Add new files to the directory.*
* ***Delete Files:*** *Remove files from the directory.*
* ***Listing Directory:*** *View the contents of the directory (list all files).*
* ***Updating Directory:*** *Modify the directory (e.g., adding or removing files).*

***2. Naming:***

* ***Users need to refer to files by name.***
* ***Unique Filenames:*** *Each file must have a unique name, but users may not know all the filenames in a system.*
* ***Tree Structure:*** *The directory follows a* ***tree structure****, allowing users to find a file by following the directory path.*
* ***Duplicate Filenames:*** *Duplicate filenames are allowed if they have* ***different pathnames*** *(e.g., "file1" in two different folders).*

***3. File Sharing:***

*In a* ***multiuser system****, files can be shared among users, but there are two key concerns:*

1. ***Access Rights:*** *Define who can do what with the file.*
2. ***Management of Simultaneous Access:*** *Ensures that multiple users can access the file at the same time without conflicts.*

***4. Access Rights:***

*Various systems implement different* ***access rights****, often in a hierarchy where one right implies others. Common access rights include:*

* ***None:*** *User may not know the file exists.*
* ***Knowledge:*** *User knows the file exists and who the owner is.*
* ***Execution:*** *User can load and execute the file but cannot copy it.*
* ***Reading:*** *User can read the file (including copying and execution).*
* ***Appending:*** *User can add data but cannot modify or delete existing data.*
* ***Updating:*** *User can modify, delete, and add data to the file.*
* ***Changing Protection:*** *User can change access rights for others.*
* ***Deletion:*** *User can delete the file.*

***5. User Classes:***

*Users are classified based on their access to files:*

1. ***Owner:*** *Typically, the file creator with full access rights.*
2. ***Specific Users:*** *Rights are explicitly granted to individual users.*
3. ***User Groups:*** *A set of users sharing the same rights.*
4. ***All:*** *Everyone has access.*

***6. Simultaneous Access:***

* ***File Locking:***
  + *A user may lock an entire file when updating it.*
  + *A user may also lock individual records during an update.*
* ***Issues in Simultaneous Access:***
  + ***Mutual Exclusion:*** *Ensures only one user can modify the file at a time.*
  + ***Deadlock:*** *Occurs when users block each other, preventing further progress.*

***Security in Operating Systems***

***Historical Background***

* *Early personal computers (PCs) like MS-DOS had minimal security because they were intended for single-user, isolated use.*
* *Physical protection, such as locking the room where PCs and floppy disks were kept, was the primary form of security.*
* *With the advent of networking and shared systems, the need for robust software-based security increased dramatically.*

***Key Security Concepts***

1. ***Confidentiality***
   * *Ensures data or services are protected from unauthorized access.*
   * *Example: Encrypting sensitive documents.*
2. ***Integrity***
   * *Ensures data or services are delivered as intended without unauthorized modification.*
   * *Example: Digital signatures prevent data tampering.*
3. ***Availability***
   * *Ensures the system remains accessible for legitimate users.*
   * *Example: Protecting servers against Distributed Denial of Service (DDoS) attacks.*

***The Security Problem***

* ***Secure System****: Resources are used as intended under all circumstances.*
* ***Threat****: A potential violation of security.*
* ***Attack****: An attempt to breach security, either malicious or accidental.* 
  + ***Malicious Attack****: Deliberate hacking or malware.*
  + ***Accidental Attack****: Unintentional errors leading to system compromise.*

***Types of Threats***

1. ***Breach of Confidentiality***
   * *Unauthorized access to sensitive data.*
   * *Example: Reading private emails.*
2. ***Breach of Integrity***
   * *Unauthorized modification of data.*
   * *Example: Changing bank records.*
3. ***Breach of Availability***
   * *Destruction or obstruction of services.*
   * *Example: Server crash due to excessive traffic.*
4. ***Theft of Service***
   * *Unauthorized use of system resources.*
   * *Example: Using someone else’s cloud account.*
5. ***Denial of Service (DoS)***
   * *Preventing legitimate use of resources.*
   * *Example: Overloading a server with requests.*

***Common Penetration Techniques***

1. ***Password Exploits****:*
   * *Guessing passwords or using vendor-default passwords.*
   * *Example: Users setting weak passwords like “12345.”*
2. ***Phishing Attacks****:*
   * *Fake login screens to steal user credentials.*
   * *Example: A website mimicking a banking portal.*
3. ***Malware****:*
   * *Malicious software like viruses, worms, and Trojan horses.*
   * *Example: A Trojan horse disguised as legitimate software.*

***Types of Threats in Detail***

* ***Active Threats****:* 
  + *Directly damage systems or data.*
  + *Example: Ransomware encrypting files.*
* ***Passive Threats****:* 
  + *Focus on gathering data without altering the system.*
  + *Example: Eavesdropping on network traffic.*

***Specific Examples of Threats***

1. ***Accidental Errors****:*
   * *Deleting a critical file by mistake.*
   * *Solution: Regular backups and recovery procedures.*
2. ***Malicious Misuse****:*
   * *Trojans, viruses, or unauthorized tampering.*
   * *Example: A Trojan stealing financial data.*
3. ***Backdoors****:*
   * *Secret entry points left by developers.*
   * *Risk: Unauthorized users exploiting them.*
4. ***Logic Bombs****:*
   * *Triggers based on conditions (e.g., a specific date).*
   * *Example: A program deleting files on April 1.*
5. ***Trojan Horse****:*
   * *Appears useful but hides harmful code.*
   * *Example: A free software installer with hidden malware.*

***Malware and Its Impact***

1. ***Viruses****:*
   * *Infect programs or files and propagate.*
   * *Example: A macro virus infecting Word documents.*
2. ***Worms****:*
   * *Spread independently over networks.*
   * *Example: Email worms sending themselves to all contacts.*
3. ***Types of Virus Propagation****:*
   * ***Append****: Attaching malicious code to legitimate programs.*
   * ***Replace****: Overwriting original files with malicious ones.*
   * ***Redirect****: Altering the control flow to execute malicious code.*
4. ***Advanced Viruses****:*
   * ***Polymorphic Virus****: Changes code to evade detection.*
   * ***Stealth Virus****: Hides its presence from antivirus software.*

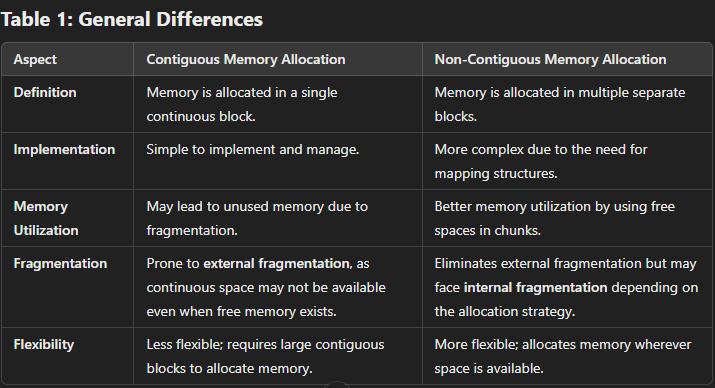
***Mitigation Strategies***

1. ***Encryption****:*
   * *Secure data in transit and storage.*
2. ***Regular Updates****:*
   * *Keep systems patched against vulnerabilities.*
3. ***Antivirus Software****:*
   * *Detect and remove malicious programs.*
4. ***User Education****:*
   * *Prevent phishing and weak password practices.*
5. ***Access Controls****:*
   * *Limit permissions based on user roles.*

***IMP To Note***

*A black screen with white text

Description automatically generated*

**

*A screenshot of a computer

Description automatically generated*

*A black screen with white text

Description automatically generated*

*A screenshot of a computer screen

Description automatically generated*

***File Management in Operating Systems***

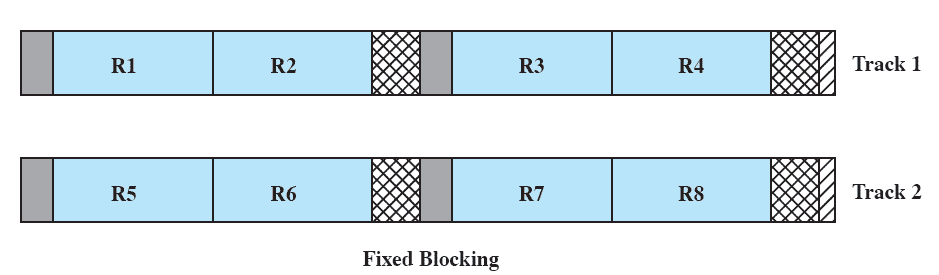
*File management involves organizing, storing, retrieving, and updating data on secondary storage. It ensures efficient use of storage while providing a way for users and programs to access files reliably.*

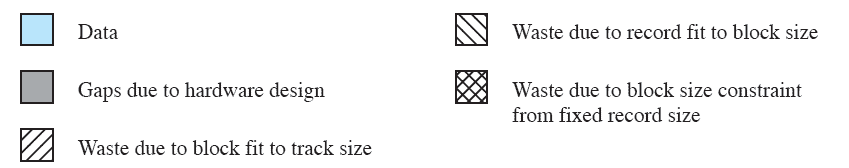
***1. Record Blocking***

*When data (records) are stored, they are grouped into* ***blocks*** *to facilitate I/O operations. Each block is the smallest unit read or written to the storage.*

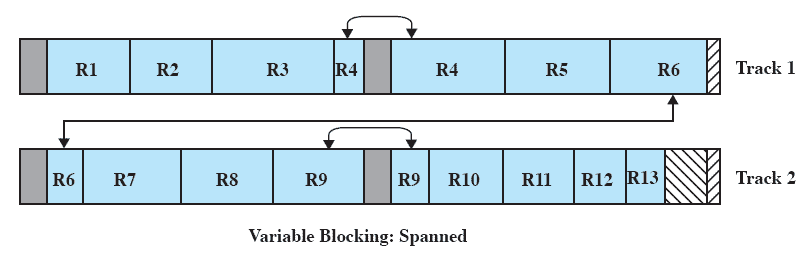
***Types of Blocking:***

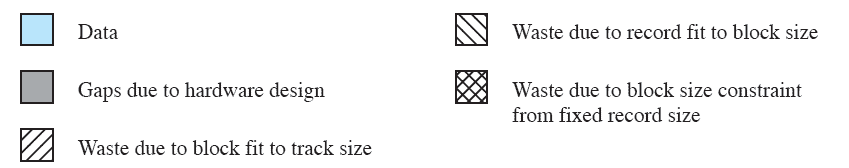
1. ***Fixed-Length Blocking:***
   * *Each block contains records of the same size.*
   * *Simple to manage but can leave unused space in a block if records don't fit perfectly.*
   * ***Problem:******Internal Fragmentation*** *– leftover space inside blocks goes unused.*



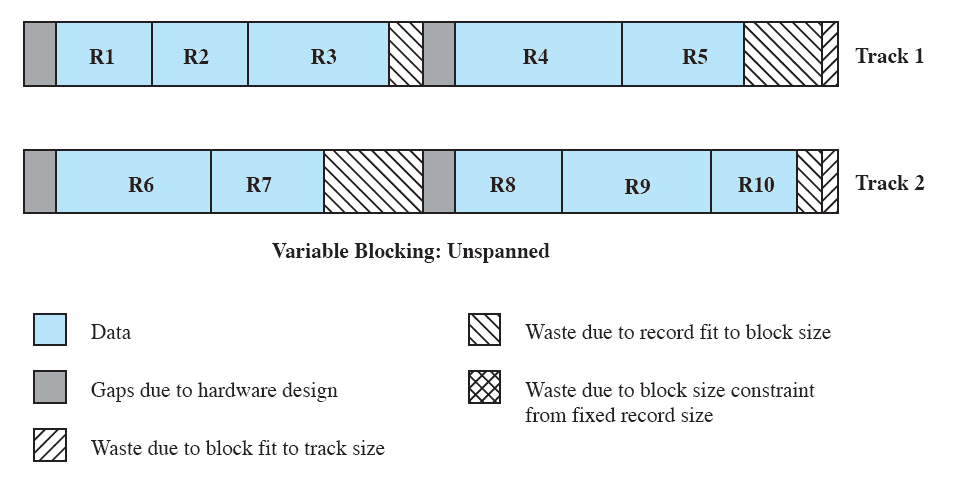


1. ***Variable-Length Blocking (Spanned):***
   * *Records of varying sizes are stored, and if a record doesn't fit in one block, it "spans" into the next block.*
   * *No unused space but adds complexity in managing records.*





1. ***Variable-Length Blocking (Unspanned):***
   * *Records of varying sizes are stored, but no single record spans across multiple blocks.*
   * *Leaves leftover space in blocks where records don't fit.*



***2. Secondary Storage Management***

*Secondary storage (like hard drives or SSDs) needs efficient allocation techniques to manage files.*

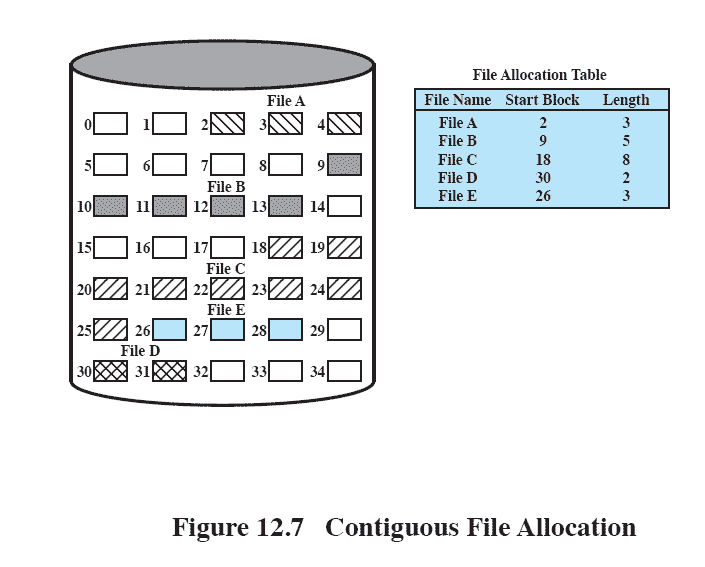
***Key Allocation Methods:***

1. ***Preallocation:***
   * *Reserves a fixed amount of space for a file when it's created.*
   * *If the reserved space isn’t fully used, it leads to* ***wastage.***
   * ***Advantage:*** *Reduces the risk of running out of space while writing the file.*
2. ***Dynamic Allocation:***
   * *Allocates space as the file grows.*
   * *Avoids wasted space but is harder to manage because the file might end up scattered across the disk.*

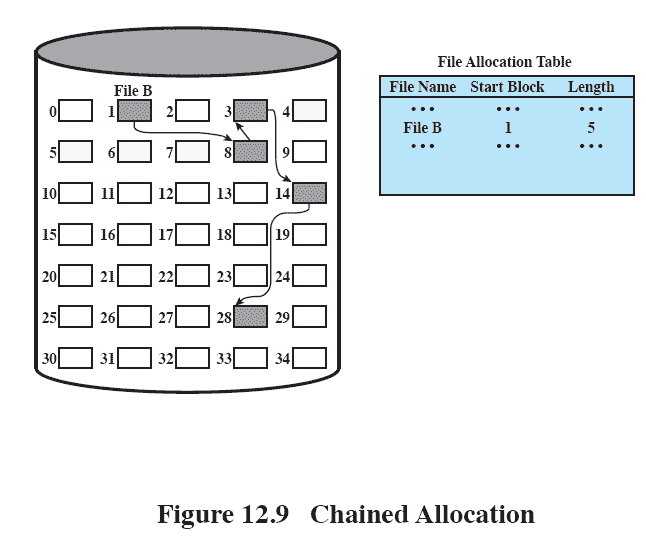
***3. File Allocation Methods***

*The way file data is allocated to disk blocks determines performance and efficiency:*

1. ***Contiguous Allocation:***
   * *Stores the entire file in consecutive blocks on the disk.*
   * ***Advantages:***
     + *Best performance for sequential access.*
     + *Easy to implement; reduces seek time during file reads.*
   * ***Disadvantages:***
     + ***External Fragmentation:*** *As files are deleted and new ones are written, free spaces become scattered.*
     + *Need for* ***compaction*** *to reorganize and merge free spaces.*



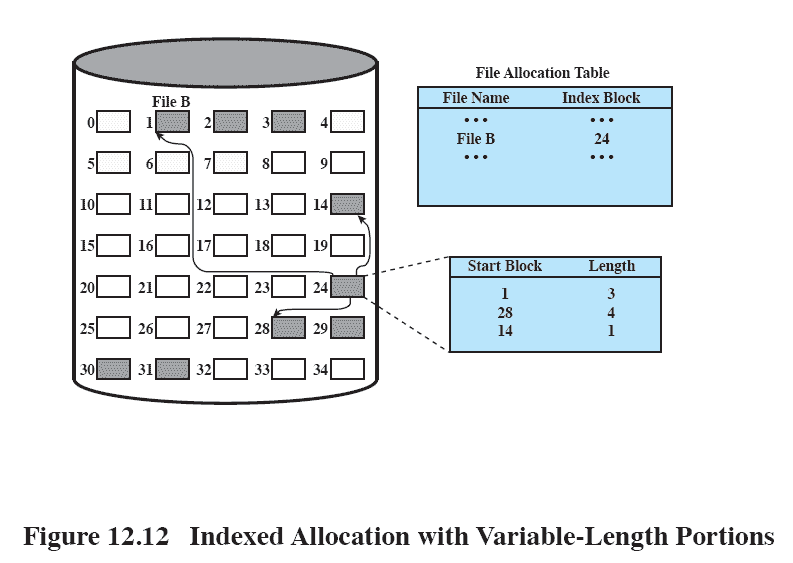
1. ***Chained Allocation:***
   * *Each block contains a pointer to the next block of the file.*
   * ***Advantages:***
     + *No need for contiguous free blocks; avoids external fragmentation.*
     + *Good for files that grow over time.*
   * ***Disadvantages:***
     + *Slow for random access since you need to follow the chain.*
     + *Overhead due to pointers stored in blocks.*



1. ***Indexed Allocation:***
   * *Uses an index (a table) to keep track of all the blocks of a file.*
   * ***Advantages:***
     + *Allows random access directly to any block.*
     + *No external fragmentation.*
   * ***Disadvantages:***
     + *Extra space needed to store the index.*
     + *If the index table becomes large, accessing it can slow down file operations.*

A diagram of a block diagram

Description automatically generated



***4. Free Space Management***

*The system must track unused blocks to allocate them efficiently when files are created or expanded.*

***Common Techniques:***

1. ***Bit Tables:***
   * *Each bit represents a block: 0 means free, 1 means allocated.*
   * *Compact and efficient but requires scanning to find free blocks.*
2. ***Chained Free Portions:***
   * *Free blocks are linked in a chain, with each pointing to the next.*
   * *Easy to manage but slower for finding large contiguous free spaces.*
3. ***Indexing:***
   * *Free space is managed like a file, with an index keeping track of free blocks.*
   * *Useful for systems that already rely on indexed allocation.*
4. ***Free Block List:***
   * *A simple list of free blocks.*
   * *Easy to update but inefficient for quickly finding free space.*

***5. Fragmentation***

*Fragmentation occurs when storage space isn't utilized efficiently, leading to wasted space.*

1. ***Internal Fragmentation:***
   * *Occurs when allocated blocks are larger than needed.*
   * *Example: A file requiring 2 KB is stored in a 4 KB block, wasting 2 KB.*
2. ***External Fragmentation:***
   * *Happens when free blocks are scattered across the disk, making it hard to allocate contiguous space.*
   * *Example: Multiple small free spaces exist, but none are large enough for a new file.*

***Solutions to Fragmentation:***

* *Use* ***chained allocation*** *or* ***indexed allocation*** *to avoid the need for contiguous space.*
* *Perform* ***compaction*** *to consolidate scattered free blocks into larger continuous spaces.*
* *Design filesystems with flexible allocation strategies to minimize wasted space.*

***6. File Allocation Examples***

* ***Contiguous Allocation:*** *Example – A file stored sequentially from block 100 to block 120.*
* ***Chained Allocation:*** *Example – File stored at block 100, with pointers linking it to blocks 200 and 300.*
* ***Indexed Allocation:*** *Example – A table lists block numbers: 100, 150, 250.*

***Advantages and Disadvantages of Allocation Methods***

| ***Allocation Method*** | ***Advantages*** | ***Disadvantages*** |
| --- | --- | --- |
| *Contiguous* | *Fast for sequential access; simple to manage* | *Suffers from external fragmentation* |
| *Chained* | *Flexible file growth; no external fragmentation* | *Slow random access; space wasted by pointers* |
| *Indexed* | *Efficient random access; avoids fragmentation* | *Needs extra space for the index* |