ASSIGNMENT

Operating Systems

By Shiyas A

Contents

File organisation directory structure along with blocking & buffering in OS.

In an operating system (OS), the file organization directory structure refers to the way files and directories are arranged and managed on a storage medium, such as a hard disk. It provides a hierarchical representation of files and directories, allowing users and applications to easily access and organize their data. The most common type of directory structure is the tree-like structure, where directories can contain subdirectories, and files are stored within the directories.

The typical components of a file organization directory structure are:

1. Root Directory: The top-most directory in the hierarchy, represented as "/". It does not have a parent directory.

2.Directories (Folders): Containers used to group related files and subdirectories together. They can have a name and can be nested within other directories.

3.Files: Data stored on the storage medium, such as documents, images, executables, etc. They are contained within directories.

4.Path: A unique sequence of directory names that specifies the location of a file or directory in the directory structure. For example, "/home/user/documents/report.txt" is a path that leads to the "report.txt" file located in the "documents" directory under the "user" directory.

The directory structure makes it easier to organize and manage files, providing a logical organization and preventing naming conflicts between files with the same name by placing them in separate directories.

Blocking and Buffering in OS:

1.Blocking:

Blocking is a technique used in operating systems to manage input/output (I/O) operations involving large amounts of data, such as reading or writing data to a storage device (e.g., hard disk). Instead of performing individual reads or writes for each small piece of data, the OS groups data into larger blocks or chunks and processes them together. This has several advantages:

- Reduced Overhead: By reading or writing data in larger blocks, the overhead associated with individual I/O requests is reduced. The OS deals with fewer requests, resulting in better overall performance.

- Improved Efficiency: When data is read or written in blocks, the OS can optimize the process by reordering or rearranging the I/O operations, minimizing seek time and reducing data fragmentation on the storage medium.

- Better Throughput: Blocking allows for better data transfer rates, as more data can be processed in a single operation compared to handling each data unit separately.

2.Buffering:

Buffering is a mechanism used to temporarily store data during I/O operations to improve efficiency and performance. The OS allocates a portion of the memory as a buffer, which acts as an intermediate storage area between the application and the storage device.

When data is read from the storage device, it is first placed in the buffer. This way, if the application needs to read the same data again or requires additional data from the same block, the OS can directly retrieve it from the buffer without having to perform another I/O operation. Similarly, when data is written, it is first placed in the buffer, and the OS manages the writing process to the storage device at an appropriate time.

Buffering offers several advantages:

- Reduced I/O Operations: By buffering data, the number of actual I/O operations to the storage device can be minimized, reducing overhead and improving overall performance.

- Latency Hiding: Since data is readily available in the buffer, the application can access it quickly, hiding the latency associated with accessing the slower storage device.

- Sequential I/O Optimization: Buffering allows the OS to optimize sequential reads and writes, which can significantly improve performance for tasks involving large sequential data access.

UNIX File structure

The UNIX file structure, also known as the UNIX filesystem hierarchy, is the organized framework used by the UNIX operating system and its derivatives (such as Linux) to manage and store files, directories, and system-related information on storage devices. This structure is hierarchical, with various directories (folders) and files arranged in a specific manner to facilitate efficient management, navigation, and access.

Key elements of the UNIX file structure:

1.Root Directory (/): The top-level directory in the hierarchy, denoted by a single forward slash (/). All other directories and files are organized under the root directory.

2.Directories (Folders): Directories are used to organize and group related files together. Each directory can contain files and subdirectories, creating a tree-like structure. Directories can also be referred to as folders.

3.Files: Files contain data and can include various types of information, such as text, programs, scripts, configurations, and more. Files are organized within directories.

4.Path: A path is a string that specifies the location of a file or directory within the file structure. There are two types of paths:

5.Absolute Path: Specifies the full path from the root directory to the desired file or directory (e.g., /home/user/documents/file.txt).

Relative Path: Specifies the path relative to the current working directory (e.g., ../folder/file.txt).

Special Directories:

/bin: Contains essential system binaries (executable files).

/etc: Stores system configuration files.

/home: User home directories are typically located here.

/tmp: Temporary files are stored in this directory.

/var: Contains variable data, such as log files, spool directories, and cached content.

6.Mount Points: UNIX systems can have multiple storage devices (hard drives, partitions, etc.). Each storage device can be attached to the filesystem at a specific mount point (directory). This allows users and applications to access data on different devices as if they were part of the same hierarchy.

7.Device Files: Special files located in the /dev directory that represent hardware devices, such as hard drives, printers, and network interfaces. These files provide a way for user programs to interact with hardware.

8.Symbolic Links: Symbolic links (symlinks) are files that point to other files or directories. They enable you to create shortcuts or references to other locations in the filesystem.

9.Permissions and Ownership: Each file and directory in the UNIX file structure has associated permissions that control who can read, write, and execute them. Files and directories also have owners and groups, determining which users have control over them.

10.Filesystem Navigation and Manipulation Commands: UNIX provides various commands for navigating and manipulating the filesystem, such as cd (change directory), ls (list files), mkdir (create directory), cp (copy), mv (move or rename), rm (remove), and more.

The UNIX file structure is designed to be organized, flexible, and efficient, allowing users and system administrators to manage and access files and directories in a consistent and logical manner.

How is Secondary Storage managed?. What is swap-space used for?

Secondary storage management refers to the management of storage devices that are used to store data outside of the computer's main memory (RAM). This typically includes hard disk drives (HDDs), solid-state drives (SSDs), and other forms of external storage. Secondary storage is used to store data and programs that are not currently being actively processed by the CPU.

Some key aspects of secondary storage management:

1.File System Management: The operating system organizes and manages data on secondary storage devices using a file system. A file system provides a way to organize, store, and retrieve files and directories. It handles tasks such as file creation, deletion, reading, writing, and access control.

2.Disk Scheduling: When multiple processes or applications are trying to access the storage device simultaneously, the operating system uses disk scheduling algorithms to determine the order in which requests are serviced. This helps optimize the use of the storage device and reduce access time.

3.Caching: Caching involves storing frequently accessed data in a faster storage medium (such as RAM) to improve data access times. This helps reduce the need to access slower secondary storage for frequently used data.

4.Virtual Memory and Paging: Virtual memory is a technique that allows an operating system to use secondary storage as an extension of RAM. When the available physical RAM is insufficient to hold all the running processes, the operating system swaps out less-used portions of memory to secondary storage. This process is known as paging. Swap space is a designated portion of the secondary storage used for paging. It acts as a safety net to prevent the system from running out of memory entirely.

Swap Space:

- Swap space, also known as swap space or paging file, is a reserved area on a secondary storage device (usually a hard drive or SSD) that the operating system uses as an extension of physical memory (RAM).

- When the available physical RAM is fully utilized, the operating system transfers some data from RAM to the swap space to free up memory for other tasks.

- Swap space is used as a temporary storage area for data that is not actively being used, allowing the system to maintain the illusion of having more available memory than physically installed RAM.

- The data in swap space is typically moved back into RAM when it's needed, though accessing data from swap space is slower than accessing data from RAM due to the slower nature of secondary storage.

- Having too much reliance on swap space can lead to performance degradation, as accessing data from secondary storage is significantly slower than accessing data from RAM.

- Properly managing swap space is important to ensure that the system remains responsive and efficient. If a system frequently relies heavily on swap space, it might indicate that there is a shortage of physical RAM, and adding more RAM could improve performance.

How is access and security managed for Distributed systems?

Access Management in Distributed Systems:

1.Authentication: In distributed systems, authentication involves verifying the identity of users or entities before granting access. This is achieved through methods like exchanging digital tokens, usernames and passwords, cryptographic keys, or biometric data. The system uses these credentials to determine whether a user is genuine and permitted to access resources.

2.Authorization: After authentication, the system checks what actions or data the authenticated user is allowed to access. This involves using access control lists (ACLs), policies, and role-based access control (RBAC) mechanisms to determine whether a user has the necessary permissions to perform specific operations.

3.Encryption: Data encryption ensures that information exchanged between distributed nodes remains confidential and secure. This involves encrypting data using algorithms and keys, making it unreadable to unauthorized parties. Secure communication protocols like TLS/SSL are used to encrypt data during transmission.

4.Secure Communication: Distributed systems utilize secure communication protocols to establish encrypted channels between nodes. This prevents unauthorized interception, eavesdropping, or tampering of data as it travels between different parts of the system.

Security in Distributed Systems:

In a distributed system, one must consider many possible security risks. To mitigate these risks there are a number of strategies that can be employed:

- Encryption algorithms that protect data in transit and at rest.
- Firewalls that limit access to specific ports/cables.
- Intrusion detection systems that identify anomalous behavior among network services.
- Intrusion prevention systems (IPS) respond to attempted intrusions by initiating defensive actions like blocking suspicious IP addresses or taking down compromised servers.

These measures may be insufficient, to identify attacks at the network level without help from other sources. We can not only prevent malicious actors from gaining access to our machines from other machines in the same firewall but can also monitor our own actions.

Reckless data sharing can significantly increase exposure to both the threats themselves and the costs entailed in defending against them.

Goals of Distributed System Security:

Security in a distributed system poses unique challenges that need to be considered when designing and implementing systems. A compromised computer or network may not be the only location where data is at risk; other systems or segments may also become infected with malicious code. Because these types of threats can occur anywhere, even across distances in networks with few connections between them, new research has been produced to help determine how well distributed security architectures are actually performing**.**

How do you manage remote file systems?

Managing remote file systems involves accessing, organizing, and controlling files and directories that reside on a different computer or server over a network. This process is essential for sharing and collaborating on data across distributed environments.

how remote file systems are managed:

1. Network File Sharing Protocols:

Remote file systems are typically accessed using network file sharing protocols such as:

- Server Message Block (SMB): Commonly used in Windows environments, SMB allows users to access shared files and printers over a network. It also supports authentication, authorization, and access control.
- Network File System (NFS): Primarily used in Unix-like systems, NFS enables remote access to files and directories as if they were on a local system. It supports centralized authentication and export controls.

2. Mounting Remote File Systems:

To manage remote file systems, you need to "mount" them, which means making the remote files and directories accessible as if they were part of your local file system. This process involves connecting to the remote server and creating a link between the remote directory and a local mount point.

- Mount Command: In Unix-like systems, you use commands like mount or mount.cifs (for SMB) to establish the connection and map the remote file system to a local directory.
- Drive Mapping: In Windows, you can map network drives to assign a drive letter to a remote file share, making it accessible from the local file explorer.

3. File System Navigation and Operations:

Once the remote file system is mounted, you can navigate and perform file operations (e.g., create, read, write, delete) on remote files and directories using standard file manipulation commands, just as you would with local files.

4. Network Latency and Performance:

It's important to note that accessing remote file systems introduces network latency, which can affect file access and data transfer speeds. Slower network connections might result in slower file operations, so optimizing the network and using efficient protocols can improve performance.

5. Security and Access Control:

Managing security in remote file systems involves:

- Authentication: Users need valid credentials (username and password) to access remote shares. Kerberos and Active Directory integration provide secure authentication.
- Authorization: Remote file systems support access control mechanisms like ACLs (Access Control Lists) and permissions to determine who can read, write, or modify files.
- Encryption: Network file sharing protocols often support encryption to secure data during transmission.

6. Disconnection and Unmounting:

When you're done using the remote file system, you should properly disconnect or unmount it. Failing to unmount can lead to data integrity issues. Use commands like umount (Unix-like systems) or simply disconnect network drives (Windows).

7. Backup and Recovery:

Regularly backing up remote file systems is crucial to ensure data recovery in case of accidental deletion, corruption, or hardware failure.

8. Automation and Scripting:

For managing large-scale or repetitive tasks on remote file systems, automation and scripting tools (e.g., Bash, PowerShell) can be used to streamline operations and improve efficiency.

Managing remote file systems requires an understanding of the specific file sharing protocol being used, proper authentication and access control practices, and considerations for network performance and security.

Circuit Switching :

- Circuit Switching is done by setting a physical path between two systems.
- In circuit switching, data is not stored
- Circuit Switching needs a dedicated physical path that's why the messages need not be addressed.
- Circuit Switching is Geographical addressing.
- Circuit Switching is costlier.
- Circuit switching routing is manual type routing.
- Circuit switching reserves the full bandwidth in advance. because of that, there is a lot of wastage of bandwidth.
- In-circuit switching, the charge depends on time and distance.
- Congestion occurs per minute in circuit switching.
- Circuit switching uses Analog and digital media on a variety of platforms.
- In-circuit switching there is no propagation delay.
- The transmission capacity of circuit switching is very low.
- In a circuit switching, Messages need not be addressed as there is one dedicated path.
- Ex. Real time transfer of voice signals.

Message Switching :

- In message Switching, data is first stored by one node then forward to another node to transfer the data to another system.
- In message Switching, data is first stored, then forwarded to the next node.
- Message switching does not need a dedicated physical path and on Message switching, The messages are addressed independently.
- Message Switching is Hierarchical addressing.
- The cost of message switching is less than circuit switching.
- Message Switching routing is not manual type routing, here route is selected during call setup.
- Message Switching does not reserve the entire bandwidth in advance. and that's why bandwidth is used to its maximum extent.
- In message switching, the charge is based on the number of bytes and distance.
- In message switching, no congestion or very little congestion occurs.
- Whereas Message switching uses digital media on a variety of platforms.While In Message switching there is a propagation delay.
- while the transmission capacity of message switching is high.
- In message switching, Messages are addressed as independent routes are established.
- Ex. Transmission of telegram.

Packet switching :

- Packet switching is a communication method where data is divided into smaller units called packets and transmitted over the network.
- Each packet contains the source and destination addresses, as well as other information needed for routing.
- The packets may take different paths to reach their destination, and they may be transmitted out of order or delayed due to network congestion.
- Efficient use of bandwidth: Packet switching is efficient because bandwidth is shared among multiple users, and resources are allocated only when data needs to be transmitted.
- Flexible: Packet switching is flexible and can handle a wide range of data rates and packet sizes.
- Scalable: Packet switching is highly scalable and can handle large amounts of traffic on a network.
- Lower cost: Packet switching is less expensive than circuit switching because resources are shared among multiple users.
- Higher latency: Packet switching has higher latency than circuit switching because packets must be routed through multiple nodes, which can cause delay.
- Limited QoS: Packet switching provides limited QoS guarantees, meaning that different types of traffic may be treated equally.
- Packet loss: Packet switching can result in packet loss due to congestion on the network or errors in transmission.