OPERATING SYSTEM [G]

Individual Assignment

Zubair Ahmed

19-39745-1

TASK-1

As the company deals with secured information from the different branches throughout the country, they need a strong, reliable, and secure open-source Operating System. An Open-Source Operating System is one where the source code is publicly accessible and modifiable. From the common list of open-source Operating Systems, I would like to suggest **Linux OS.** Its open-source feature enables developers to create patch updates for any vulnerabilities discovered before cybercriminals can attack them. Users favor Linux OS because it is a stable operating system that is clear of malware attacks, reducing the concern of having to replace the operating system due to malware infestation.

Fundamentals of the Linux Operating System:

- Application separation is a methodology for preventing any program from interacting with other programs that are already running.
- The number of resources that an application or account can consume in its functions is restricted by resource allocation control.
- The only root account that can perform all administrative functions is a user account.
- The Linux kernel's Netfilter framework is used by the System Firewall to control network communications.
- Secure remote access with OpenSSH implements a secure encryption technology and provides the user direct access to the following domains,

    i. Command-line access for remote access,
    ii. Execution of a distant command,
    iii. Transferring files,
    iv. Access to graphical software from a distance.

- Encrypted storage is used to encrypt storage containing sensitive data. To access an encrypted disk, users must first supply the decryption password.
- In the case of an attack, host integrity testing is used to validate the integrity of a running system.

Reference: 10 Most Secure Operating Systems (#9 Is Our Favourite) 🔒🔑 (secureblitz.com)

TASK-2

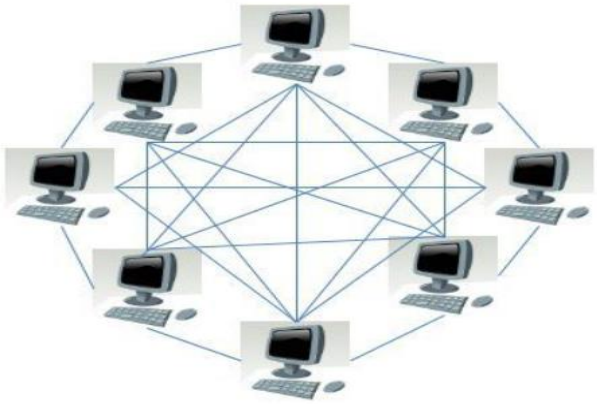
CLIENT-SERVER NETWORK:

In Client-Server Network there is a centralized server which stores the data as its management is centralized. In this network, the server responds to the clients requested services.

Client-Servers Network Model

PEER-TO-PEER NETWORK:

In Peer-to-Peer Network, every node has the ability to both request and respond to services. This model does not distinguish between clients and servers; instead, each node acts as both a client and a server.



Peer-to-Peer Network Model

Difference between **Client-Server Network** and **Peer-to-Peer Network**

Client-Server Network	**Peer-to-Peer Network**
1. Clients and servers are differentiated in a client-server network, and certain servers and clients are present.	**1.** Clients and servers are not differentiated in a peer-to-peer network.
2. The data is stored on a centralized server in a Client-Server Network.	**2.** Each peer in a peer-to-peer network has its own data.
3. Information sharing is the foundation of a Client-Server Network.	**3.** The Peer-to-Peer Network, on the other hand, is concerned with connectivity.
4. In a client-server network, the server responds to the services that the client has requested.	**4.** Every node in a peer-to-peer network can request and response to services.
5. A server may become severely limited if multiple customers request services at the same time.	**5.** A server is not seriously limited since the services are dispersed among numerous servers in the peer-to-peer system.
6. Client-Server architecture is more dependable and scalable.	**6.** If the number of peers in the system grows, Peer-to-Peer degrades.
7. The client-server system is expensive to develop.	**7.** Peer-to-peer networks are less costly to build.

References:

1. Difference between Client-Server and Peer-to-Peer Network - GeeksforGeeks

2. Difference Between Client-Server and Peer-to-Peer Network (with Comparison Chart) - Tech Differences

TASK-3

API: The API (Application Programming Interface) is a software interface that allows two programs to communicate with one another and work together.

When we are using a smartphone application, it connects to the Internet and sends information to a server. The server then receives the information, interprets it, takes the appropriate actions, and sends it back to our phone. The software simply analyses the data and displays the information we requested in an accessible form. All of these happened through API.

System Call: System Call is the programmatic way in which a computer program requests a service from the kernel of the operating system it is executed on. The interface between a process and the operating system is provided via a system call.

A file open is a system call in which we specify the file's path and the system function responds with a handle that allows us to control it even further.

Explanation of how API and System Call works together to perform a task

System call exposes the operating system's services to user programs through an API (Application Programming Interface). The kernel system can only be accessed using system calls.

The system-call interface detects API function calls and conducts the operating system's required system calls. Each system call is often assigned a number, and the system-call interface keeps a table indexed by these numbers. The system call interface then calls the specified system call in the kernel of the operating system and returns the status of the call as well as any return values. The caller just needs to follow the API and comprehend what the operating system will do as a result of the system call's execution. As a result, the API hides most of the specifics of the operating-system interface from the programmer and is controlled by the run-time support library. When a user program calls the open() system function, the operating system handles it.

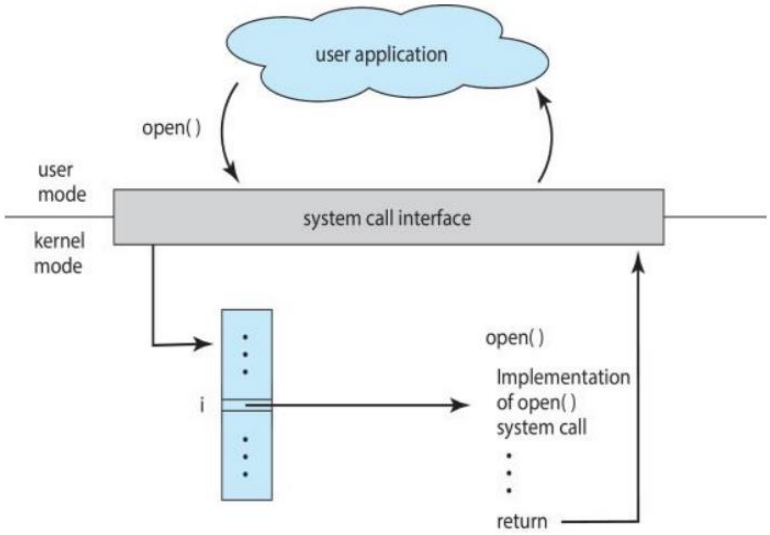


Figure 1.13 The handling of a user application invoking the open() system call.

TASK-4

Layered structure is a system structure in which the operating system's numerous services are divided into layers, with each layer performing a distinct, well-defined task.

In an operating system, layering has a specific advantage. Each layer can be defined independently and interact with one another as needed. In addition, creating, maintaining, and updating a system in the form of layers is easy. A change in one layer's specification has no bearing on the other layers.

The layered operating system is made up of six levels,

1. **Hardware:**

   This layer communicates with the system hardware and manages the operation of all peripherals such as a printer, mouse, keyboard, scanner, and so on. The hardware layer is responsible for managing certain types of hardware devices.

2. **CPU Scheduling:**

   This layer is in charge of scheduling CPU processes. Processes are managed using a variety of scheduling queues. The processes are placed in the job queue as soon as they enter the system.

3. **Memory Management**

   Memory management is concerned with memory and the movement of processes from disk to primary memory for execution, as well as the reverse. The operating system's third layer is in charge of this. This layer is responsible for all memory management. In a computer, different types of memories exist, such as RAM and ROM.

4. **Process Management**

   This layer is in charge of managing processes, such as allocating processors to processes and determining how many processes will remain in the waiting queue. In this layer, the processes' priority is also managed.

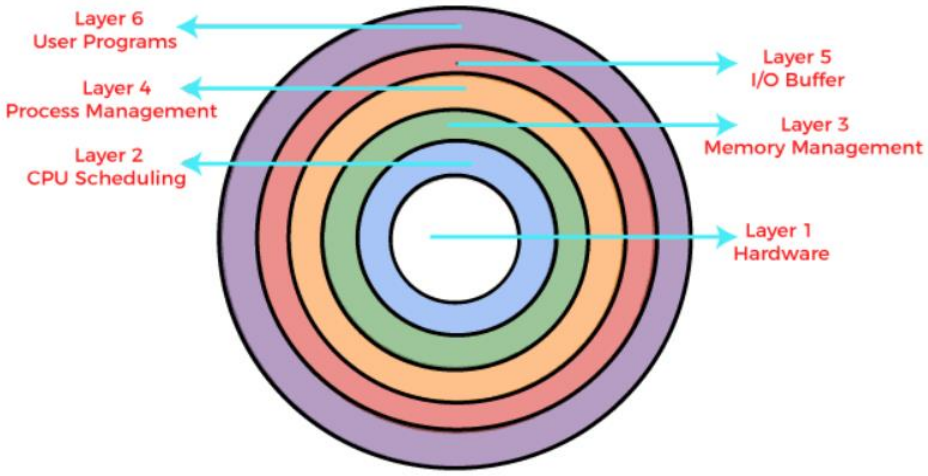
5. **I/O Buffer**

   In computer systems, I/O devices are critical. They give users the ability to engage with the system. This layer is in charge of the buffers for the I/O devices and ensures that they function properly.

6. **User Programs**

   In a layered operating system, this is the topmost layer. This layer is responsible for the numerous user programs and applications that run on an operating system, including as word processors, games, and browsers.

   Through following these layered structure an OS of a game device can be designed.



Reference: Layered Structure of Operating System - javatpoint