Day 1: Linux Systems

Task 1:

Kernel Architecture Diagram - Draw a detailed diagram of the Linux kernel architecture.

Label and write a short description (2-3 sentences) for each major component like Scheduler, File System, Network Stack, etc.

<table>
<tr><td colspan="2">User Space</td></tr>
<tr><td>User Applications</td><td>System Libraries</td></tr>
<tr><td colspan="2">System Call Interface</td></tr>
<tr><td colspan="2">Kernel Space</td></tr>
<tr><td>Process Management</td><td>Memory Management</td></tr>
<tr><td>Scheduler</td><td>Virtual Memory</td></tr>
<tr><td>Inter-Process Communication (IPC)</td><td>Physical Memory Management</td></tr>
<tr><td>File Systems</td><td>Network Stack</td></tr>
<tr><td>Device Drivers</td><td>Architecture Dependent Code</td></tr>
<tr><td colspan="2">Hardware</td></tr>
</table>

User Space:

- **User Applications:** Programs and applications run by the users.
- **System Libraries:** Libraries like glibc that provide standard functionalities to applications and interact with the kernel via system calls.

System Call Interface:

- A boundary between user space and kernel space. It provides a set of interfaces (system calls) for applications to interact with the kernel, requesting services such as I/O operations, process management, etc.

Kernel Space:

- **Process Management:**

    - **Scheduler:** Manages the execution of processes, determining which process runs at any given time, ensuring efficient CPU usage and process prioritization.
    - **Inter-Process Communication (IPC):** Mechanisms for processes to communicate and synchronize with each other, including signals, pipes, message queues, shared memory, and semaphores.
- **Memory Management:**
    - **Virtual Memory:** Provides an abstraction of memory, giving each process the illusion of having a large, contiguous address space. It handles paging and swapping.
    - **Physical Memory Management:** Manages the actual physical memory (RAM), tracking which parts are free, allocated, and handling memory allocation and deallocation.
- **File Systems:**
    - Manages the storage and retrieval of data on disk drives. It provides a hierarchical structure of directories and files, and supports multiple file system types like ext4, XFS, and NTFS.
- **Network Stack:**
    - Implements networking protocols (TCP/IP stack) to handle data transmission over network interfaces, providing functionalities for socket communication, packet routing, and network device management.
- **Device Drivers:**
    - Abstractions that allow the kernel to communicate with hardware devices. Each type of hardware (e.g., storage devices, network adapters, graphics cards) has its corresponding driver.
- **Architecture Dependent Code:**
    - Contains code specific to the hardware architecture (e.g., x86, ARM) for low-level operations such as context switching, interrupt handling, and I/O operations.

Task 2:

Use command-line tools to perform the following tasks in Linux:

Create a directory structure /tmp/myfiles and navigate into it.

Create three text files, write sample content in them, and list the files in the directory.

Change the permissions of one file to read-only for the group.