Multi-Client Remote Command Execution System using C++

(Socket and Process-based Server-Client Model)

In modern networked environments, system administrators often need to execute and monitor commands on remote machines without physically accessing them. Simultaneously managing several clients, ensuring process isolation, and achieving responsiveness without blocking resources can prove to be a challenge for administrators. Traditional tools like SSH solve this at a high level, but understanding their underlying mechanisms requires a deep grasp of operating system concepts such as process creation, inter-process communication (IPC), and socket programming.

Proposed Solution / Project Idea:

This project implements a mini remote shell system using C++ on Linux environment, where a server accepts various client connections simultaneously for the execution of commands from the clients.

- The server accepts client connections and forks a separate child process for each client connection using the system call fork().
- Each client sends a shell command (like ls -l) to the server.
- The server will perform the command in a grandchild process by using execvp(), capture the command output through a pipe, and send it back to the client by using TCP sockets.
- The client program provides an interactive prompt for sending commands and displays the results received from the server.

The setup demonstrates multi-process handling, communication between different processes, and networking, mimicking how real remote command execution systems function internally.

Course Topics Covered:

- Process Management
- Inter-Process Communication (IPC)
- Concurrency & Synchronization
- Networking (Sockets API)
- Error Handling & Signal Management
- System calls