National University of Computer & Emerging Sciences
Karachi Campus

Bank Transaction Processing System

Project Proposal
Operating Systems
BCS-4D

Group Members:
23I-0710 | Hasan Sami
23K-0812 |Sameel Jamal
23K-0837 | Raahim Irfan

Project Proposal

- **Introduction**

The Bank Transaction Processing System is a multi-threaded and multi-process application that simulates concurrent banking operations such as deposit, withdrawal, and account transfers. The primary aim is to demonstrate Operating System-level synchronization using PThreads, mutexes, and semaphores to manage shared data and avoid race conditions.

- **Existing System**

In real-world banking environments, core banking solutions such as Oracle FLEXCUBE and Temenos T24 process thousands of transactions simultaneously. These systems ensure data consistency, atomicity of transactions, and concurrency control. However, such systems are highly complex and operate behind proprietary architectures. Our project serves as a simplified prototype to model similar concurrency challenges using PThreads.

- **Problem Statement**

Most beginner-level banking simulators do not simulate true concurrent processing or synchronization issues like race conditions or deadlocks. Furthermore, they often lack realistic concurrency mechanisms for managing multiple client operations or teller availability. This limits students' understanding of actual OS-level process/thread management and synchronization.

- **Proposed Solution**

Our solution introduces true multi-threaded and multi-process interaction between banking entities. Each transaction is handled in a separate thread, with mutexes ensuring exclusive access to shared account data and semaphores used to simulate limited resources (e.g., available ATMs or bank tellers). This system will demonstrate realistic concurrency control and atomic operations.

- **Salient Features**
    - Concurrent deposits, withdrawals, and transfers using threads
    - Mutex-based synchronization for shared account access
    - Semaphore-controlled ATM/teller availability
    - Simulation of race conditions and resolution
    - Transaction logging for auditing purposes
    - Realistic modeling of delays and waiting queues
    - Deadlock and starvation avoidance
    - User interface via terminal/CLI

- **Tools & Technologies**
    - Programming Language: C/C++
    - Libraries: POSIX Threads (PThreads), POSIX Semaphores
    - Operating System: Linux (Ubuntu preferred)

- Tools: GCC/G++, Terminal, Makefile