OS Project

Thread Management Simulator

Rawan sous

Rivan jaradat

Dana ismail

Project description:

Thread Management Simulator

Implement a thread management system that allows the creation, execution, and

synchronization of multiple threads. Include functionalities such as thread creation,

termination, synchronization primitives (e.g., locks, semaphores), and context switching.

In the Thread Management component of the operating system simulator, you will

implement a system that handles the creation, execution, and synchronization of multiple

threads.

Programming language used :python

Workflow:

We divided the step to three parts, each will be assigned to manage one part, but also communicate and

Debug concurrently to insure the proper working of the functionalities and the algorithms of tasks or processes into the system.

Implement the mechanism to generate a sequence of tasks or processes with their associated arrival times and burst times.

Role 1: Thread Creation and Termination

Responsibilities:

Implement the interface or command for creating new threads.

Handle the specification of entry point functions for each thread.

Develop mechanisms for thread termination, such as explicit termination requests or thread completion detection.

Role 2: Thread Execution and Context Switching

Responsibilities:

Implement the scheduler that determines the order of thread execution.

Simulate the execution of threads by executing their associated functions or code blocks.

Implement context switching between threads to allow each thread to make progress and share the CPU.

Role 3: Synchronization and Communication

Responsibilities:

Implement synchronization primitives like locks, semaphores, or condition variables.

Create mechanisms for thread synchronization, such as thread signaling, waiting, and notifying.

Enable communication and data sharing between threads through shared memory or message passing.