Nick Johnson and Rakin Reza

Operating Systems

Professor Roya Hosseini

Project 2

For this project, we worked on simulating process execution using threads and implementing synchronization with the Producer-Consumer problem. The main goal was to understand how operating systems manage processes and control access to shared resources between threads.

First, we created threads from the `processes.txt` input file, where each line represents a process with its ID and burst time. Each process is simulated as a separate thread, and to represent CPU bursts, we used `Thread.sleep()` based on the burst time. When a thread starts and finishes, it prints a message so we can follow the process execution in real-time.

After setting up basic threading, we moved on to synchronization. We chose the Producer-Consumer problem because it's a classic example of thread coordination. In our implementation, a producer thread adds items to a bounded buffer, and a consumer thread removes them. We used Java's BlockingQueue class, which internally handles synchronization and prevents overproduction or under-consumption automatically. Also, a synchronized block was used to avoid race conditions when multiple threads access the buffer at the same time.

Throughout the simulation, messages are printed to show when threads are waiting, producing, consuming, or finishing. This made it easier to debug and also clearly shows how the

synchronization is working. Testing with the given input file helped make sure that the code could handle multiple processes and random execution orders without running into issues like deadlock or starvation.

Overall, this project helped us better understand how threads work and why synchronization is so important. Even though the Producer-Consumer problem looks simple at first, actually coding it made us realize how easy it is for things to go wrong without proper locks and control. This experience gave us a much clearer picture of what real operating systems have to deal with under the hood when managing multiple processes and resources.