Faculty of Sciences and Engineering
Department of Computer Science
CSC411 - Integrative Programming Technologies

Producer-Consumer Problem Implementation Report

Group Members:
Lungelo Dlamini (202203681)
Michael Mamba (202202335)

This report documents the comprehensive implementation of the Producer-Consumer problem, successfully addressing all requirements outlined in the mini project specification. The project demonstrates robust concurrent programming principles through two distinct architectures: a threaded version utilizing shared memory with bounded buffers and semaphores, and a socket-based version employing network communication for distributed processing. Both implementations effectively handle the core challenge of synchronizing data production and consumption while maintaining data integrity and system stability.

The threaded implementation forms the foundation of our solution, featuring three synchronized components working in harmony. The Producer thread generates realistic student data by randomly selecting from predefined lists of names, programmes, and courses, then creates XML files following the naming convention student1.xml through student10.xml. Each file contains complete student information including randomly generated 8-digit student IDs and course marks between 0-100. The heart of our synchronization mechanism lies in the Bounded Buffer class, which implements a thread-safe queue with a maximum capacity of 10 elements using semaphores for empty/full condition tracking and mutex locks for critical section protection. This ensures the producer never adds data when the buffer is full, and the consumer never attempts to remove data when the buffer is empty. The Consumer thread retrieves file numbers from the buffer, processes the corresponding XML files by calculating student averages, determining pass/fail status based on the 50% threshold, displaying comprehensive results on screen, and finally deleting the processed files to maintain system cleanliness.

Our XML handling implementation demonstrates sophisticated data serialization and deserialization capabilities. The ITStudent class serves as the data model, featuring methods for converting objects to XML strings and reconstructing objects from XML files. The XML structure carefully mirrors the required student information, with proper nesting of courses and marks within the document hierarchy. The system maintains a shared directory between producer and consumer where XML files are created, processed, and subsequently removed, ensuring efficient resource management and preventing file accumulation. The consumer's output provides clear, formatted displays showing student names, IDs, programmes, courses with individual marks, calculated averages, and final pass/fail determinations, giving complete visibility into the processing pipeline.

The socket programming implementation extends our solution to distributed systems, demonstrating network-based producer-consumer communication. This version employs a client-server architecture where the socket producer acts as a server listening on port 9009, and the socket consumer connects as a client. We implemented a robust communication protocol using length-prefixed messages to ensure reliable data transmission, where each XML payload is preceded by a 4-byte length indicator. The socket producer generates student data and transmits XML content over the network, while the socket consumer receives, parses, and processes the data without requiring physical file operations. This approach showcases alternative inter-process communication methods and provides scalability for distributed deployments.

Key technical achievements include:

- Proper synchronization using semaphores and mutex locks preventing race conditions
- Comprehensive error handling for file operations, buffer access, and network communication
- Configurable parameters for buffer size, production/consumption rates, and network settings
- Resource cleanup ensuring no file leaks or memory issues
- Graceful thread termination and connection management

The project successfully meets all functional requirements while demonstrating software engineering best practices. Both implementations handle edge cases effectively, including buffer boundary conditions, file access conflicts, and network interruptions. The codebase maintains clear separation of

concerns, with dedicated modules for buffer management, student data handling, producer logic, and consumer operations. This modular design facilitates maintenance and future enhancements while providing a solid educational example of concurrent programming principles in practice.

Through this implementation, we have created a robust system that not only solves the classical producer-consumer problem but also provides practical insights into real-world scenarios involving data processing, synchronization, and inter-process communication. The project stands as a comprehensive demonstration of the theoretical concepts covered in the course, brought to life through practical application and careful attention to implementation details.

The following is link to our github account: https://github.com/202203681/producer-consumer-project/