Operating Systems Lab Assignment: Concurrency with Semaphores, Critical Sections, and Monitor Locks

Your Name

Sep 17, 2025

1 Introduction

This report documents the implementations and analyses for the concurrency lab assignment, covering Producer-Consumer in C, Bank Account in C++, and additional exercises on synchronization mechanisms.

2 Exercise 1: Producer-Consumer in C

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define BUFFER_SIZE 5
#define NUM_ITEMS 10

int buffer[BUFFER_SIZE];
int in = 0, out = 0;

sem_t full, empty;
pthread_mutex_t mutex;

void *producer(void *arg) {
    for (int i = 0; i < NUM_ITEMS; i++) {
        int item = rand() % 100;    // Produce an item

        sem_wait(&empty);           // Wait if buffer is full
        pthread_mutex_lock(&mutex); // Lock the buffer

        buffer[in] = item;
        printf("Produced:␣%d\n", item);
        in = (in + 1) % BUFFER_SIZE;

        pthread_mutex_unlock(&mutex); // Unlock buffer
        sem_post(&full);              // Signal that one more item is available

        sleep(1); // Simulate time taken to produce
    }
    return NULL;
}

void *consumer(void *arg) {
    for (int i = 0; i < NUM_ITEMS; i++) {
        sem_wait(&full);             // Wait if buffer is empty
        pthread_mutex_lock(&mutex); // Lock the buffer
```