CSC 4320 Operating Systems

Project 02: Readers Writers Synchronization Problem

Deepak Srinivas Govindarajan

November 15, 2025

1 Overview

This project implements the Readers Writers synchronization problem using POSIX threads. The design supports concurrent reading, exclusive writing, and limits writer starvation.
Repository: `https://github.com/1DeepakSrinivas/projects-4320`

2 Implementation Details

2.1 Synchronization Primitives

Locks and Condition Variable:

- `count_mutex`: protects reader counter `r_count`.
- `mutex`: ensures exclusive writer access.
- `w_cond`: coordinates waiting writers.

Shared State:

- `shared_data`: integer critical section value.
- `r_count`: number of active readers.
- `w_waiting`: number of writers queued.

2.2 Algorithm Design

Readers increment `r_count` under `count_mutex`. The first reader locks `mutex`, blocking writers. Subsequent readers enter concurrently. The last reader unlocks `mutex` and signals waiting writers.

Writers increment `w_waiting`, then wait on `w_cond` until `r_count == 0`. Once admitted, a writer locks `mutex`, updates `shared_data`, finishes, and signals waiting threads.

This enforces concurrent reads, exclusive writes, and limits writer starvation.

2.3 Thread Configuration

- `NUM_READERS = 5`
- `NUM_WRITERS = 3`
- Reading time: 2 s
- Writing time: 3 s

Readers perform three read cycles. Writers perform two write cycles. Thread arguments include thread ID and type.

2.4 Code Architecture

The file `readers_writers.c` implements:

- `start_read()`, `end_read()`
- `start_write()`, `end_write()`
- `reader()` and `writer()` thread routines

The main control function creates and joins all threads, then destroys all synchronization primitives.

3 Results

Concurrent Reading: Multiple readers access `shared_data` simultaneously. Logs confirm identical values for overlapping read intervals.

Exclusive Writing: Writers obtain exclusive access using `mutex`. Readers pause until writer completion. `shared_data` increments sequentially with no race conditions.

Lock Coordination: Logs include thread creation, waiting, acquisition, critical section activity, and lock release. Execution shows no deadlocks or starvation.

4 Files

`output.txt` and `processes.txt` are included in `project-02/src`.

5 Conclusion

The project demonstrates proper synchronization for the Readers Writers problem using pthread mutexes and condition variables. The modular structure supports concurrent reads, exclusive writes, and fairness for writers.