Implementing Readers-Writers Solutions using Semaphores

Overview

This README document covers two C++ programs, `rw-CO22BTECH11006.cpp` and `frw-CO22BTECH11006.cpp`, designed to solve the Readers-Writers problem. The first solution prioritizes writers, while the second aims for a fairer approach between readers and writers. Both implementations use semaphores for synchronization to ensure that multiple readers can read concurrently without a writer writing and that writers have exclusive access when writing.

Structure

The project directory is organized into several subdirectories:

- `Code Files`: This directory holds the source code for the Readers-Writers problem solutions.
  - `rw-CO22BTECH11006.cpp`: Implementation with writer preference.
  - `frw-CO22BTECH11006.cpp`: Fair solution that aims to balance reader and writer access.
- `Input Files`: This folder contains the input parameters for the programs.
  - `input.txt`: Configuration file with parameters for the reader and writer threads.
- `Output Files`: Contains the output logs and statistical data generated by the programs.
  - `RW-log.txt`: Log file detailing the operation of the `rw-CO22BTECH11006` program.
  - `RW-Average_time.txt`: Statistics on the average and worst-case waiting times for the writers and readers in `rw-CO22BTECH11006`.
  - `FairRW-log.txt`: Log file for the `frw-CO22BTECH11006` program, with a fair scheduling approach.
  - `FairRW-Average_time.txt`: Statistical analysis of the average and worst-case waiting times for the `frw-CO22BTECH11006` program.

Features

- Two different approaches to solving the Readers-Writers problem.
- Use of semaphores for synchronization.
- Measurement of wait times for performance evaluation.
- Detailed logging of thread activity for analysis.

Requirements

- A modern C++ compiler (e.g., g++, clang).
- POSIX Threads library (`pthread`).
- Linux/Unix environment for semaphore and threading support.

Compilation and Execution

First navigate to the directory containing the code files and compile and run the programs using the following commands in the terminal:

For **`rw-CO22BTECH11006.cpp`**:

```
g++ -o rw-CO22BTECH11006 rw-CO22BTECH11006.cpp -lpthread
./rw-CO22BTECH11006
```

For **`frw-CO22BTECH11006.cpp`**:

```
g++ -o frw-CO22BTECH11006 frw-CO22BTECH11006.cpp -lpthread
./frw-CO22BTECH11006
```

Input File Format

Ensure the input file **`../Input Files/input.txt`** (In Input Files directory) exists and is correctly formatted, as both programs read parameters from it.

The input file should contain six space-separated values in the following order:

1. **`nw`** - Number of writer threads.
2. **`nr`** - Number of reader threads.
3. **`kw`** - Number of entries to the critical section per writer.
4. **`kr`** - Number of entries to the critical section per reader.
5. **`muCS`** - Average time in the critical section (milliseconds).
6. **`muRem`** - Average time in the remainder section (milliseconds).

Example **`input.txt`** content:

```
5 5 10 10 2000 1000
```

This indicates 5 writers and 5 readers, each attempting to enter the critical section 10 times, with an average time of 2 seconds in the critical section and 1 second in the remainder section.

Output

Each program generates two files:

1. A detailed log file (**`RW-log.txt`** for **`rw-CO22BTECH11006.cpp`** or **`FairRW-log.txt`** for **`frw-CO22BTECH11006.cpp`**) containing timestamps of critical section requests, entries, and exits for both readers and writers.
2. A statistics file (**`RW-Average_time.txt`** for **`rw-CO22BTECH11006.cpp`** or **`FairRW-Average_time.txt`** for **`frw-CO22BTECH11006.cpp`**) providing average and worst-case wait times.

Contact Information

- **Author:** Om Dave
- **Roll No.:** CO22BTECH11006

For queries or contributions, please contact the author directly.