CSE 344

Homework #4

Due Date: 12 May 23:59

This assignment will be tested and run on Debian 12 (64-bit) in VirtualBox.
Identical or highly similar submissions will receive -100 points.
A report with screenshots is mandatory. No report = 0 points.
Late submissions will not be accepted.

Scenario: You will develop a **Multithreaded Log File Analyzer** in C using **POSIX threads**, **mutexes**, **condition variables**, and **barriers**. The program will read a large log file in parallel, search for a user-defined keyword, and report matching lines. The work will be divided among worker threads, while the main thread (manager) coordinates loading data and synchronizes via a shared buffer.

Requirements

1. Thread Roles & Structure (20 points)

- Manager thread reads the input file line by line, placing them into a shared buffer.
- Worker threads consume lines from the buffer and search for the given keyword.
- Synchronization must be achieved with:
  - pthread_mutex_t for buffer locking
  - pthread_cond_t for wait/signal mechanisms
  - pthread_barrier_t to synchronize worker threads before final reporting
- Use a bounded buffer. If the buffer is full, manager must wait. If empty, workers must wait.
- Manager adds a special "EOF marker" to signal end of processing.
- You are free to implement the buffer using a circular queue, dynamic array, or any thread-safe structure.
- This can be implemented using a NULL pointer, a sentinel string, or a special struct flag.

2. Command-line Interface (10 points)

```
./LogAnalyzer <buffer_size> <num_workers> <log_file> <search_term>
```

Exampe:

```
./LogAnalyzer 20 4 /var/log/syslog "ERROR"
```

3. Output and Final Report (10 points)

- Each worker prints the number of matches it found.
- After all threads finish (synchronized with barrier), a summary report is printed by one thread.

4. Error Handling (10 points)

- If arguments are missing or incorrect, the program must print usage:

Usage: **./LogAnalyzer <buffer_size> <num_workers> <log_file> <search_term>**

- Handle SIGINT (Ctrl+C) to gracefully free memory and exit.
- All memory must be freed; test with valgrind.

5. Buffer Synchronization (20 points)

- Use condition variables to ensure the producer-consumer logic:
    - Manager waits when buffer is full
    - Workers wait when buffer is empty
- No busy-waiting allowed

6. Barrier Use (10 points)

- After all workers finish processing, use pthread_barrier_wait() to ensure they all reach the end before printing the summary.

7. Testing Scenario

Test your program with the following scenarios and provide uncut screenshots in the report.

1- valgrind ./LogAnalyzer 10 4 logs/sample.log "ERROR"

Buffer size: 10, Workers: 4, Check for memory leaks.

2- ./LogAnalyzer 5 2 logs/debug.log "FAIL"
3- ./LogAnalyzer 50 8 logs/large.log "404"

Submission Structure

```
StudentID_Name_Surname_HW4.7z
├── Makefile
├── StudentID_main.c
├── buffer.c / buffer.h
├── utils.c / utils.h (if needed)
├── StudentID_report.pdf (with screenshots)
```

- **Your Makefile must include make clean.**

- **Do not run your program in Makefile; only compile.**
- **Use `gcc` for compilation and ensure `-lpthread` is included.**
- **Your Makefile must contain both `make` and `make clean` targets.**

Grading (Total 100 Points)

Category	Description	Points
1. Thread Communication	Correct use of mutex + condition variables + producer-consumer logic	20
2. Barrier Synchronization	Correct and meaningful use of barrier for final sync	10
3. Command-line Interface	Argument parsing, error messages, usage output	10
4. Output & Result Reporting	Per-thread results, summary, proper printf usage	10
5. Signal Handling	Graceful exit on Ctrl+C, free memory	5
6. Valgrind Check	No memory leaks, all threads free resources	10
7. Test Scenario & Screenshots	Run all test cases, include clear screenshots in report	15
8. Code Quality & Compilation	Makefile, proper modular design, clean build	10

Additional Rules:

- Code does not compile → -100 points
- Missing Makefile or make clean → -30 points
- No report (or missing screenshots) → -100 points
- No mutex/condition/barrier → -50 points
- Using busy-wait → -30 points
- Late submissions → Not accepted

Report Format

Your report must include:

1. Title Page (Student Name, ID, Course, Homework #)
2. Introduction (Explain what the program does)
3. Code Explanation (Explain each function and how it works)
4. Screenshots (For each command tested)
5. Conclusion (Challenges faced, solutions, and final thoughts)

For your questions, please contact: **zbilici@gtu.edu.tr**