

Homework 4 Documentation

Abdullah TÜRKMEN

Student ID: 210104004072

Course: System Programming

Homework 4

May 12, 2025

1 Introduction

This program is a multithreaded log analyzer that reads lines from a log file, processes them using multiple worker threads, and searches for a specific term. It demonstrates the use of synchronization mechanisms, signal handling, and efficient thread communication.

2 Code Explanation

2.1 Buffer Management Functions

- **buffer_init:** Initializes the buffer, setting up its size, locks, and condition variables.
- **buffer_add:** Adds a line to the buffer, ensuring thread-safe access and handling full buffer scenarios.
- **buffer_get:** Retrieves a line from the buffer, ensuring thread-safe access and handling empty buffer scenarios.
- **buffer_destroy:** Cleans up the buffer by freeing memory and destroying locks and condition variables.

2.2 Log Analyzer Functions

- **manager_thread:** Reads lines from the log file and adds them to the buffer. Signals workers when done.
- **worker_thread:** Retrieves lines from the buffer and searches for the specified term. Updates match counts.
- **sigint_handler:** Handles SIGINT signals to gracefully stop the program and wake up waiting threads.
- **main:** Sets up the buffer, creates threads, and coordinates their execution. Displays the final results.

3 Screenshots

```
gcc -D LogAnalyzer_MatchBuffer -openbsd
abdu@abduPC:~/System-Programing/hw4$ ./LogAnalyzer 20 4 logs/sample.log "ERROR"
Manager thread started.
[Thread 1] Worker exiting. Matches: 3969
[Thread 0] Worker exiting. Matches: 3976
[Thread 3] Worker exiting. Matches: 4072
[Thread 2] Worker exiting. Matches: 3705
Main: All worker threads joined.
===== Summary =====
Thread 0 matches: 3976
Thread 1 matches: 3969
Thread 2 matches: 3705
Thread 3 matches: 4072
Total matches: 15722
Main: Cleanup finished. Exiting.
abdu@abduPC:~/System-Programing/hw4$
```

Figure 1: Program execution showing manager and worker threads.

```
abdu@abduPC:~/System-Programing/hw4$ ./LogAnalyzer 20 4 /var/log/syslog "ERROR"
Manager thread started.
^C
Received SIGINT, signaling stop...
[Thread 1] Worker exiting. Matches: 15
[Thread 3] Worker exiting. Matches: 18
[Thread 2] Worker exiting. Matches: 20
[Thread 0] Worker exiting. Matches: 15
Main: All worker threads joined.
===== Summary =====
Thread 0 matches: 15
Thread 1 matches: 15
Thread 2 matches: 20
Thread 3 matches: 18
Total matches: 68
Main: Cleanup finished. Exiting.
abdu@abduPC:~/System-Programing/hw4$
```

Figure 2: SIGINT signal handling and graceful termination.

```

ndm: Cleanup finished. Exiting.
abdu@abduPC:~/System-Programming/hw4$ valgrind ./LogAnalyzer 10 4 logs/sample.log "ERROR"
==22195== Memcheck, a memory error detector
==22195== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==22195== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==22195== Command: ./LogAnalyzer 10 4 logs/sample.log ERROR
==22195==
Manager thread started.
[Thread 0] Worker exiting. Matches: 4123
[Thread 2] Worker exiting. Matches: 3840
[Thread 1] Worker exiting. Matches: 3963
[Thread 3] Worker exiting. Matches: 3796
Main: All worker threads joined.
===== Summary =====
Thread 0 matches: 4123
Thread 1 matches: 3963
Thread 2 matches: 3840
Thread 3 matches: 3796
Total matches: 15722
Main: Cleanup finished. Exiting.
==22195==
==22195== HEAP SUMMARY:
==22195==    in use at exit: 0 bytes in 0 blocks
==22195==   total heap usage: 79,562 allocs, 79,562 frees, 7,230,414 bytes allocated
==22195==
==22195== All heap blocks were freed -- no leaks are possible
==22195==
==22195== For lists of detected and suppressed errors, rerun with: -s
==22195== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

```

Figure 3: Program Execution with Valgrind

4 Conclusion

4.1 Challenges Faced

- Implementing thread synchronization and avoiding deadlocks.
- Handling SIGINT signals without leaving threads in a waiting state.

4.2 Solutions

- Used condition variables and mutexes to synchronize thread access to the buffer.
- Ensured all threads are woken up and terminated gracefully upon receiving a SIGINT signal.

4.3 Final Thoughts

This project provided valuable experience in multithreading, synchronization, and signal handling. The implementation successfully meets the requirements and handles edge cases effectively.