HUSNIDDIN RAMAZANOV / 200104004802

CSE344 / HW4

1. Introduction

This report documents the implementation of a Multithreaded Log File Analyzer in C. The program uses POSIX threads, mutexes, condition variables, and barriers to efficiently search through log files for specified keywords. The work is divided among multiple worker threads, with a manager thread coordinating data loading through a shared buffer, implementing a producer-consumer pattern.

2. Code Explanation

Main Program (main.c)

The main program initializes the necessary data structures, creates worker and manager threads, and handles cleanup once processing is complete. It includes:

- Command-line argument parsing
- Signal handling for graceful exit on SIGINT
- Buffer initialization
- Thread creation and synchronization
- Resource cleanup

```c
int main(int argc, char *argv[]) {
    int buffer_size, num_workers;
    char *log_file, *search_term;

    // Parse command line arguments
    if (parse_args(argc, argv, &buffer_size, &num_workers, &log_file, &search_term) != 0) {
        print_usage(argv[0]);
        return EXIT_FAILURE;
    }

    // Set up signal handler for SIGINT
    setup_signal_handler();

    // Initialize the buffer
    buffer_t buffer;
    if (buffer_init(&buffer, buffer_size) != 0) {
        fprintf(stderr, "Failed to initialize buffer\n");
        return EXIT_FAILURE;
    }

    // Set up barrier for worker synchronization
    pthread_barrier_t barrier;
    if (pthread_barrier_init(&barrier, NULL, num_workers) != 0) {
        perror("Failed to initialize barrier");
        buffer_destroy(&buffer);
        return EXIT_FAILURE;
    }

    // Create worker thread data
    worker_data_t *worker_data = malloc(num_workers * sizeof(worker_data_t));
    if (!worker_data) {
        perror("Failed to allocate worker data");
        pthread_barrier_destroy(&barrier);
        buffer_destroy(&buffer);
        return EXIT_FAILURE;
    }
```

Buffer Implementation (buffer.c/buffer.h)

The buffer component implements a thread-safe circular buffer that enables communication between the manager and worker threads:

- buffer_init(): Initializes the buffer and synchronization primitives
- buffer_destroy(): Cleans up buffer resources
- buffer_put(): Producer function for adding items to the buffer
- buffer_get(): Consumer function for retrieving items from the buffer

The buffer uses mutexes and condition variables to ensure proper synchronization:

- Mutex locks prevent simultaneous access to shared data

- Condition variables signal when the buffer is not full (for producer) or not empty (for consumers)

```h
#ifndef BUFFER_H
#define BUFFER_H

#include <pthread.h>
#include <stdbool.h>

// Structure to hold a line from the log file
typedef struct {
    char *line;     // The content of the line
    bool eof_marker; // Flag to indicate end of file
} buffer_item_t;

// Circular buffer structure
typedef struct {
    buffer_item_t *items;    // Array of buffer items
    int capacity;            // Maximum buffer size
    int count;               // Current number of items
    int in;                  // Index for adding items (producer)
    int out;                 // Index for removing items (consumer)
    pthread_mutex_t mutex;   // Mutex for synchronization
    pthread_cond_t not_full; // Condition variable for not full buffer
    pthread_cond_t not_empty; // Condition variable for not empty buffer
} buffer_t;

// Initialize the buffer with given capacity
int buffer_init(buffer_t *buffer, int capacity);

// Clean up buffer resources
void buffer_destroy(buffer_t *buffer);

// Add an item to the buffer (producer)
int buffer_put(buffer_t *buffer, const char *line, bool eof_marker);

// Get an item from the buffer (consumer)
buffer_item_t buffer_get(buffer_t *buffer);

#endif /* BUFFER_H */
```

Utilities (utils.c/utils.h)

The utilities module provides supporting functions:

- Signal handling setup
- Command-line argument parsing
- Worker thread function that searches for keywords
- Manager thread function that reads the log file

The worker threads use a barrier for synchronization after processing to ensure all results are ready before the final summary is printed.

```
9
10  // Structure to hold worker thread data
11  typedef struct {
12      int thread_id;                // Worker ID
13      const char *search_term;      // Term to search for
14      buffer_t *buffer;             // Shared buffer
15      int match_count;              // Number of matches found by this worker
16      pthread_barrier_t *barrier;   // Barrier for synchronization
17  } worker_data_t;
18
19  // Structure to hold manager thread data
20  typedef struct {
21      const char *filename;         // Log file path
22      buffer_t *buffer;             // Shared buffer
23      int num_workers;              // Number of worker threads
24  } manager_data_t;
25
26  // Global flag for handling SIGINT
27  extern volatile sig_atomic_t keep_running;
28
29  // Setup signal handler for SIGINT
30  void setup_signal_handler(void);
31
32  // Parse command line arguments
33  int parse_args(int argc, char *argv[], int *buffer_size, int *num_workers,
34                 char **log_file, char **search_term);
35
36  // Print usage information
37  void print_usage(const char *program_name);
38
39  // Worker thread function
40  void *worker_thread(void *arg);
41
42  // Manager thread function
43  void *manager_thread(void *arg);
44
45  #endif /* UTILS_H */
```

3. Test Screenshots

Sample.log file

```
[2025-05-10 10:00:00] INFO: Starting web server on port 80.
[2025-05-10 10:01:05] INFO: Connection received from 10.0.0.2.
[2025-05-10 10:01:07] ERROR: Failed to authenticate user guest.
[2025-05-10 10:01:12] DEBUG: Session ID 1234 created.
[2025-05-10 10:01:45] INFO: Serving page /index.html
[2025-05-10 10:01:46] 404: Page not found /about.html
[2025-05-10 10:01:47] 404: Page not found /contact.html
[2025-05-10 10:02:10] INFO: Connection received from 10.0.0.3.
[2025-05-10 10:02:11] DEBUG: Session ID 1235 created.
[2025-05-10 10:02:12] INFO: Serving page /login.html
[2025-05-10 10:02:13] FAIL: Login failed for user admin.
[2025-05-10 10:02:14] WARN: User account locked after 3 failed attempts.
[2025-05-10 10:02:30] ERROR: Database connection timeout.
[2025-05-10 10:02:35] INFO: System health check passed.
[2025-05-10 10:03:00] DEBUG: Cache cleared for user session 1234.
[2025-05-10 10:03:10] INFO: Connection closed from 10.0.0.2.
[2025-05-10 10:03:20] 404: Page not found /missing.js
[2025-05-10 10:03:30] INFO: Backup process started.
[2025-05-10 10:03:45] DEBUG: Backup file size: 2048MB
[2025-05-10 10:04:00] INFO: Backup completed successfully.
```

```
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw4$ ./log_search 10 4 sample.log "INFO"
Worker 1 found match: [2025-05-10 10:00:00] INFO: Starting web server on port 80.
Worker 0 found match: [2025-05-10 10:01:45] INFO: Serving page /index.html
Worker 3 found match: [2025-05-10 10:02:10] INFO: Connection received from 10.0.0.3.
Worker 2 found match: [2025-05-10 10:01:05] INFO: Connection received from 10.0.0.2.
Worker 3 found match: [2025-05-10 10:02:35] INFO: System health check passed.
Worker 3 found match: [2025-05-10 10:03:10] INFO: Connection closed from 10.0.0.2.
Worker 3 found match: [2025-05-10 10:03:30] INFO: Backup process started.
Worker 1 found match: [2025-05-10 10:04:00] INFO: Backup completed successfully.
Worker 0 found match: [2025-05-10 10:02:12] INFO: Serving page /login.html
Worker 3 found 4 matches
Worker 1 found 2 matches

--- Summary Report ---
Worker 0 found 2 matches
Worker 2 found 1 matches

Total matches found: 9
```

```
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw4$ ./log_search 10 4 sample.log "ERROR"
Worker 2 found match: [2025-05-10 10:01:07] ERROR: Failed to authenticate user guest.
Worker 3 found match: [2025-05-10 10:02:30] ERROR: Database connection timeout.
Worker 3 found 1 matches
Worker 1 found 0 matches

--- Summary Report ---
Worker 0 found 0 matches
Worker 2 found 1 matches

Total matches found: 2
```

```
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw4$ ./log_search 10 4 sample.log "404"
Worker 3 found match: [2025-05-10 10:01:46] 404: Page not found /about.html
Worker 2 found match: [2025-05-10 10:01:47] 404: Page not found /contact.html
Worker 0 found match: [2025-05-10 10:03:20] 404: Page not found /missing.js

--- Summary Report ---
Worker 0 found 1 matches
Worker 3 found 1 matches
Worker 1 found 0 matches
Worker 2 found 1 matches

Total matches found: 3
hussi@HUSNIDDIN:/mnt/c/Users/Husniddin/OneDrive/Desktop/systemhw/hw4$
```

4. Conclusion

In this assignment, I implemented a multithreaded log file analyzer that efficiently processes large log files by distributing the workload among multiple worker threads. The program demonstrates several key concepts of concurrent programming:

1. Producer-Consumer Pattern: The manager thread acts as the producer, reading lines from the log file and placing them in the shared buffer, while worker threads act as consumers, removing lines from the buffer and searching for keywords.
2. Thread Synchronization: Mutexes and condition variables ensure that the buffer operations are thread-safe and efficient, avoiding busy-waiting.
3. Barrier Synchronization: A barrier ensures that all worker threads complete their processing before the final summary is generated.
4. Signal Handling: The program can gracefully exit when interrupted with Ctrl+C (SIGINT).

Some challenges faced during implementation included:

- Ensuring proper cleanup of resources to avoid memory leaks
- Implementing correct synchronization to prevent deadlocks
- Handling edge cases like empty files or search terms not found

The implemented solution is efficient and scalable, capable of processing large log files by leveraging multiple threads and properly synchronizing their work.