CSE 344 System Programming – Homework 4 Report

General Structure

Initialization:

1. Parse command-line arguments.
2. Initialize the buffer with specified size.
3. Create and start manager and worker threads.

Manager Thread:

1. Traverse the source directory recursively.
2. For each file, add a FilePair (source and destination paths) to the buffer.
3. Signal worker threads when a file is added.

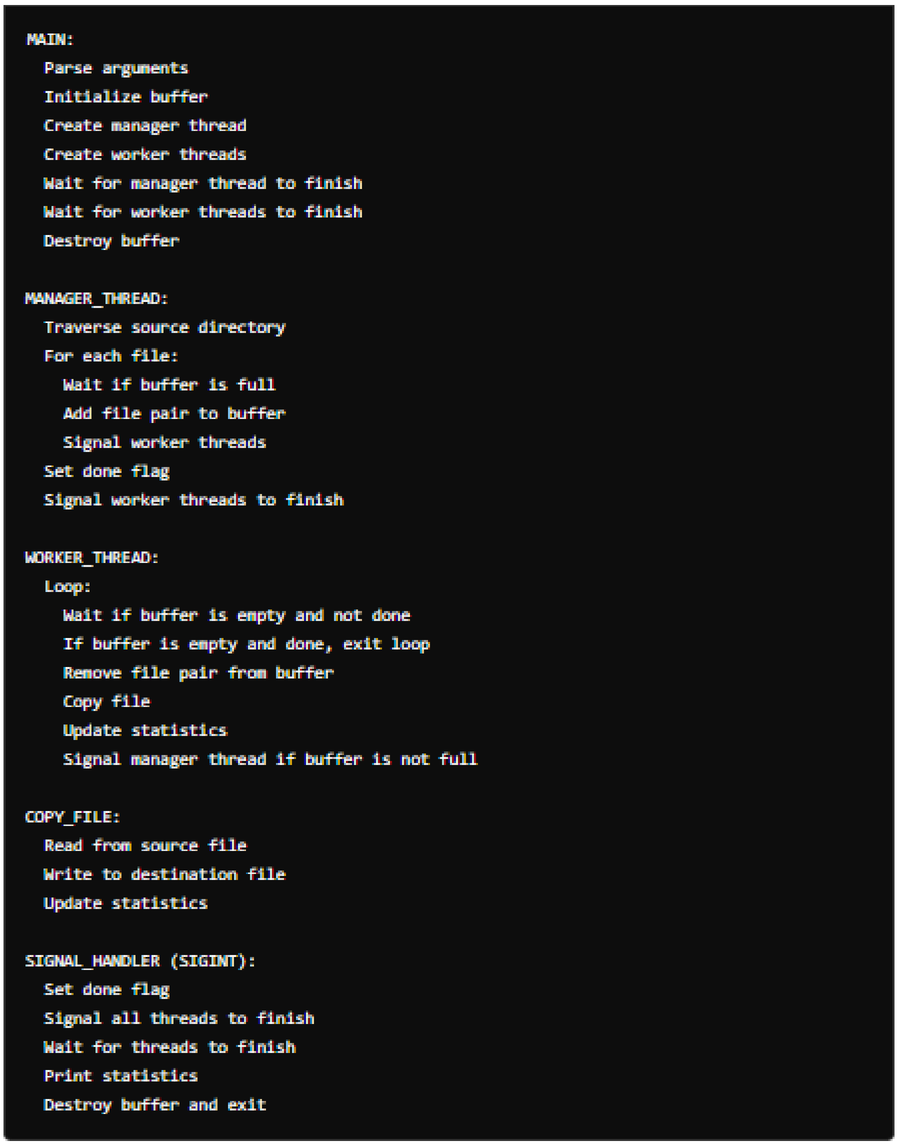
Worker Threads:

1. Wait for files to be available in the buffer.
2. Copy files from source to destination.
3. Update statistics on files copied, bytes copied, and errors encountered.

Signal Handling:

1. Handle SIGINT (CTRL+C) for graceful termination.
2. Signal all threads to finish processing and clean up resources.

Pseudocode



```
MAIN:
  Parse arguments
  Initialize buffer
  Create manager thread
  Create worker threads
  Wait for manager thread to finish
  Wait for worker threads to finish
  Destroy buffer

MANAGER_THREAD:
  Traverse source directory
  For each file:
    Wait if buffer is full
    Add file pair to buffer
    Signal worker threads
  Set done flag
  Signal worker threads to finish

WORKER_THREAD:
  Loop:
    Wait if buffer is empty and not done
    If buffer is empty and done, exit loop
    Remove file pair from buffer
    Copy file
    Update statistics
    Signal manager thread if buffer is not full

COPY_FILE:
  Read from source file
  Write to destination file
  Update statistics

SIGNAL_HANDLER (SIGINT):
  Set done flag
  Signal all threads to finish
  Wait for threads to finish
  Print statistics
  Destroy buffer and exit
```

Key Points

Thread Usage

- *Manager Thread:* Responsible for traversing directories and adding files to the buffer.
- *Worker Threads:* Multiple threads that handle copying files from the buffer to the destination directory. These threads form a thread pool.

Condition Variables

- *not_empty:* Used to signal worker threads when the buffer has new files added by the manager thread.
- *not_full:* Used to signal the manager thread when space becomes available in the buffer as worker threads remove files from it.

Synchronization: pthread_cond_wait and pthread_cond_signal are used for synchronizing access to the buffer between the manager and worker threads.

Usage in Manager Thread

- *Wait:* The manager thread waits (pthread_cond_wait) on not_full when the buffer is full, indicating it cannot add more files until space is available.
- *Signal:* After adding a file pair to the buffer, the manager thread signals (pthread_cond_signal) the not_empty condition variable to wake up worker threads waiting for new files.

Usage in Worker Threads

- *Wait:* Worker threads wait (pthread_cond_wait) on not_empty when the buffer is empty, indicating there are no files to process.
- *Signal:* After removing a file pair from the buffer, worker threads signal (pthread_cond_signal) the not_full condition variable to wake up the manager thread, indicating space is available.

Buffer Structure

- *Circular Buffer:* Implements a circular queue to store FilePair structures.
- *Critical Section:* Access to the buffer is protected by a mutex to prevent race conditions.

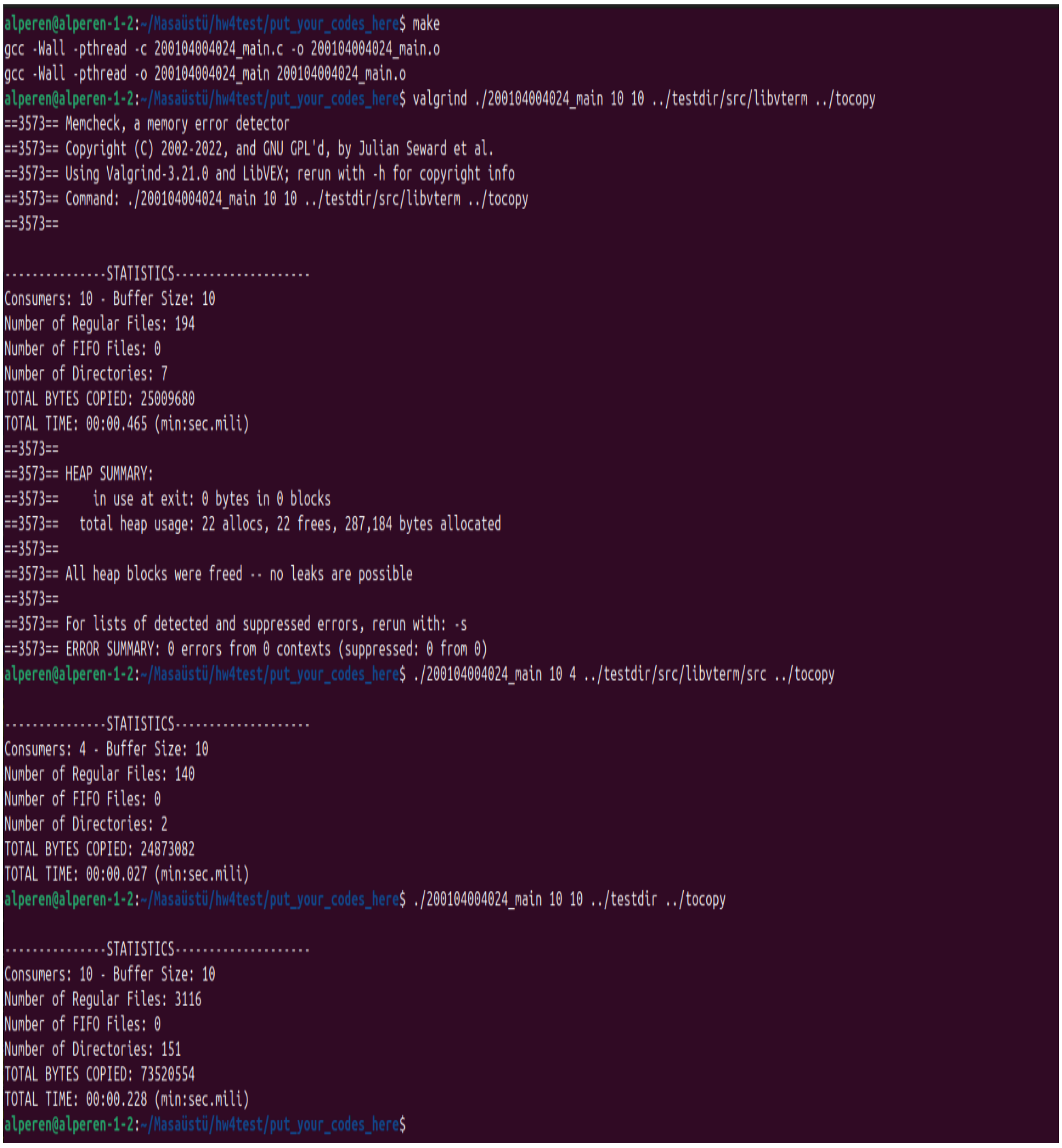
Critical Section

- *Mutex:* pthread_mutex_t is used to ensure exclusive access to shared resources like the buffer and statistics.
- *Protected Code:* All modifications to the buffer and statistics are done within critical sections to prevent data corruption.

Worker Thread Pool

- *Initialization:* Worker threads are created at the start and run concurrently.
- *Task Execution:* Each worker thread waits for a file to be available in the buffer, processes it (copies the file), and updates the statistics.
- *Termination:* Workers continue processing until all files are copied and the done flag is set by the manager thread.

Screenshots



```
alperen@alperen-1-2:~/Masaüstü/hw4test/put_your_codes_here$ make
gcc -Wall -pthread -c 200104004024_main.c -o 200104004024_main.o
gcc -Wall -pthread -o 200104004024_main 200104004024_main.o
alperen@alperen-1-2:~/Masaüstü/hw4test/put_your_codes_here$ valgrind ./200104004024_main 10 10 ../testdir/src/libvterm ../tocopy
==3573== Memcheck, a memory error detector
==3573== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.
==3573== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==3573== Command: ./200104004024_main 10 10 ../testdir/src/libvterm ../tocopy
==3573==

---------------STATISTICS-----------------
Consumers: 10 - Buffer Size: 10
Number of Regular Files: 194
Number of FIFO Files: 0
Number of Directories: 7
TOTAL BYTES COPIED: 25009680
TOTAL TIME: 00:00.465 (min:sec.mili)
==3573==
==3573== HEAP SUMMARY:
==3573==     in use at exit: 0 bytes in 0 blocks
==3573==   total heap usage: 22 allocs, 22 frees, 287,184 bytes allocated
==3573==
==3573== All heap blocks were freed -- no leaks are possible
==3573==
==3573== For lists of detected and suppressed errors, rerun with: -s
==3573== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
alperen@alperen-1-2:~/Masaüstü/hw4test/put_your_codes_here$ ./200104004024_main 10 4 ../testdir/src/libvterm/src ../tocopy

---------------STATISTICS-----------------
Consumers: 4 - Buffer Size: 10
Number of Regular Files: 140
Number of FIFO Files: 0
Number of Directories: 2
TOTAL BYTES COPIED: 24873082
TOTAL TIME: 00:00.027 (min:sec.mili)
alperen@alperen-1-2:~/Masaüstü/hw4test/put_your_codes_here$ ./200104004024_main 10 10 ../testdir ../tocopy

---------------STATISTICS-----------------
Consumers: 10 - Buffer Size: 10
Number of Regular Files: 3116
Number of FIFO Files: 0
Number of Directories: 151
TOTAL BYTES COPIED: 73520554
TOTAL TIME: 00:00.228 (min:sec.mili)
alperen@alperen-1-2:~/Masaüstü/hw4test/put_your_codes_here$
```

Alperen Duran 200104004024