CSE 344
SYSTEM PROGRAMMING

HW 5 Report

Emre Güven

1901042621

INTRODUCTION

This project is a multi-threaded file copying utility designed to traverse a source directory and replicate its structure and contents to a destination directory using a buffer and worker threads for efficient file handling. It maintains statistics on the types of files copied and the total bytes transferred.

Mutexes, condition variables and also barriers are used for synchronization of worker threads.

To compile program in shell: $> make
To run program correctly: $> ./MWCp <task_buffer_size> <#of workers> <source_folder> <destination_folder>
To clean unnecessary files: $> make clean

In explanation of used variables and structures, I indicated the changes with * at beginning of each line. In pseudo code, I indicated changes with bold and italic text.

USED MUTEXES, CONDITION VARIABLES, BARRIER

Mutexes
buffer_mutex: Protects access to the task buffer and related variables (buffer_front, buffer_rear, buffer_count).
stats_mutex: Protects access to the statistics variables (num_regular_files, num_fifo_files, num_directories, num_other_files, total_bytes_copied).
stdout_mutex: Synchronizes access to standard output.
* finished_threads_mutex: Used for safely updating and checking the count of finished threads.

Condition Variables
buffer_not_empty: Signals worker threads that the buffer has at least one task available.
buffer_not_full: Signals the manager thread that there is space available in the buffer.

* Barrier
pthread_barrier_t: The barrier ensures that all worker threads wait for each other before proceeding when they are idle and there are enough tasks.

USED TYPES AND GLOABAL VARIABLES

Types
file_task_t: A structure representing a file copying task. It holds file descriptors for the source file and destination file.
directory_paths_t: A structure representing the source and destination directories.

Global Variables
task_buffer: A dynamically allocated buffer that holds the file copying tasks to be processed by worker threads.
number_of_workers: Stores the number of worker threads that will process tasks from the task buffer.
buffer_size: Defines the size of the task buffer, determining how many tasks can be queued at any given time.
buffer_front, buffer_rear, buffer_count: Indexes and count for handling buffer.

done_flag: A flag indicating whether the copying process should be terminated, used for handling signals safely.

num_regular_files, num_fifo_files, num_directories, num_other_files: Counters for different types of files and directories processed.

total_bytes_copied: Accumulates the total number of bytes copied across all files.

* number_finish_threads: Counters for finished threads when there is signal or there is no task.

PROGRAM STRUCTURE AS PSEUDOCODE

The program is a multi-threaded file copying utility that initializes global variables and sets up necessary signal handlers. The ***main function*** parses command-line arguments, allocates memory for the task buffer, initializes a barrier for synchronizing worker threads, creates manager and worker threads, and measures the execution time. The ***manager_thread*** traverses the source directory, adds tasks to the buffer, and signals completion. ***Worker threads*** process these tasks by copying files and updating statistics.

* In the **worker threads**, after processing each task, the thread checks if there are enough remaining tasks for all threads to participate in the barrier. If there are enough tasks, the thread waits at the barrier using pthread_barrier_wait. This ensures that threads do not proceed independently, but rather synchronize with each other when idle. If a thread finishes processing and there are no more tasks, it increments the number_finish_threads count safely using the mutex and then breaks out of the loop.

Each function is explained as pseudocode.

Main function
    Parse command line arguments and check them
    Allocate memory for task_buffer and initialize it
    Call setup_signal_handler
    Initialize barrier
    Create manager thread
    Create worker threads
    Measure time
    Join manager and worker threads
    Print statistics
    Free resources and destroy mutexes and condition variables
    Destroy barrier

worker_thread function
    Loop indefinitely
        Lock buffer_mutex
        Wait for buffer_not_empty if buffer_count is 0 and done_flag is not set
        Finish loop if buffer_count is 0 and done_flag is set
            Update number_finish_threads safely by using finished_threads_mutex
            Unlock buffer_mutex
        Get task from task_buffer
        Update buffer_front and buffer_count
        Signal buffer_not_full
        Unlock buffer_mutex
        Call copy_file with current task

Lock finished_threads_mutex
If number_finish_threads is 0 and remaining tasks are enough for all threads
    Unlock finished_threads_mutex
    Wait at the barrier
Else
    Unlock finished_threads_mutex

copy_file function
Read from source_fd and write to destination_fd in chunks
Handle read/write errors
Close source_fd and destination_fd
Lock stats_mutex
Update total_bytes_copied
Unlock stats_mutex

manager_thread function
Call traverse_directory with source_folder and destination_folder
Lock buffer_mutex
Set done_flag
Broadcast buffer_not_empty and buffer_not_full
Unlock buffer_mutex

traverse_directory function
Open source_folder directory
Create destination_folder if it does not exist
For each entry in source_folder
  If entry is a directory and not "." or ".."
      Call handle_directory with source_folder, destination_folder, entry
  Else   Call handle_file with source_folder, destination_folder, entry
Close directory

handle_file function
Construct source_file and destination_file paths
Open source_file for reading
Open destination_file for writing
Create file_task_t with file descriptors and file paths
Lock buffer_mutex
Wait for buffer_not_full if buffer_count is buffer_size and done_flag is not set
Break if done_flag is set
Add task to task_buffer
Update buffer_rear and buffer_count
Signal buffer_not_empty
Unlock buffer_mutex
Call update_statistics with entry

handle_directory function
    Construct new_source and new_dest paths
    Call traverse_directory with new_source and new_dest to handle sub directory
    Lock stats_mutex
    Increment num_directories safely by using stats_mutex
    Unlock stats_mutex

update_statistics function
    Lock stats_mutex
    Update file type counters based on entry type
    Unlock stats_mutex

setup_signal_handler function
    Set signal handlers for SIGINT, SIGTERM, SIGQUIT, SIGHUP, SIGUSR1, SIGUSR2

handle_signal function
    Set done_flag
    Broadcast buffer_not_empty and buffer_not_full

destroy_mutexes_and_cond_vars function
    Destroy buffer_mutex, stats_mutex, stdout_mutex
    Destroy buffer_not_empty and buffer_not_full

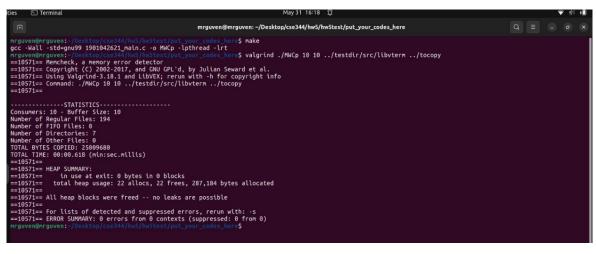**close_fd function**
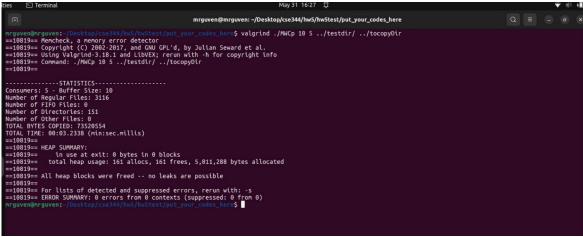   Close file descriptor and handle errors

TESTS

NOTE: Printing the completion status of each file is not shown in some test screenshots because it may takes space in report. I have given shown examples below for one short number of folder entries and one CTRL+C example. Done tests are similar to homework 4.

Copying different folders completely (No printing)

mrguven@mrguven: ~/Desktop/cse344/hw5/hw5test/put_your_codes_here

```
mrguven@mrguven:~/Desktop/cse344/hw5/hw5test/put_your_codes_here$ valgrind ./MWCp 5 8 ../testdir/runtime ../tocopyCC
==10901== Memcheck, a memory error detector
==10901== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10901== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10901== Command: ./MWCp 5 8 ../testdir/runtime ../tocopyCC
==10901==

---------------STATISTICS--------------------
Consumers: 8 - Buffer Size: 5
Number of Regular Files: 1899
Number of FIFO Files: 0
Number of Directories: 98
Number of Other Files: 0
TOTAL BYTES COPIED: 24826550
TOTAL TIME: 00:01.1074 (min:sec.millis)
==10901==
==10901== HEAP SUMMARY:
==10901==     in use at exit: 0 bytes in 0 blocks
==10901==   total heap usage: 111 allocs, 111 frees, 3,262,600 bytes allocated
==10901==
==10901== All heap blocks were freed -- no leaks are possible
==10901==
==10901== For lists of detected and suppressed errors, rerun with: -s
==10901== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
mrguven@mrguven:~/Desktop/cse344/hw5/hw5test/put_your_codes_here$
```

Copying folders completely (Printing)

mrguven@mrguven: ~/Desktop/cse344/hw5/hw5test/put_your_codes_here

```
mrguven@mrguven:~/Desktop/cse344/hw5/hw5test/put_your_codes_here$ make
gcc -Wall -std=gnu99 1901042621_main.c -o MWCp -lpthread -lrt
mrguven@mrguven:~/Desktop/cse344/hw5/hw5test/put_your_codes_here$ valgrind ./MWCp 5 8 ../testdir/src/xdiff ../tocopyF
==10956== Memcheck, a memory error detector
==10956== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10956== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10956== Command: ./MWCp 5 8 ../testdir/src/xdiff ../tocopyF
==10956==
Copied ../testdir/src/xdiff/xmacros.h to ../tocopyF/xmacros.h, 1901 bytes
Copied ../testdir/src/xdiff/xtypes.h to ../tocopyF/xtypes.h, 1529 bytes
Copied ../testdir/src/xdiff/xutils.c to ../tocopyF/xutils.c, 9553 bytes
Copied ../testdir/src/xdiff/xemit.h to ../tocopyF/xemit.h, 1204 bytes
Copied ../testdir/src/xdiff/xutils.h to ../tocopyF/xutils.h, 1779 bytes
Copied ../testdir/src/xdiff/xemit.c to ../tocopyF/xemit.c, 7848 bytes
Copied ../testdir/src/xdiff/xhistogram.c to ../tocopyF/xhistogram.c, 9638 bytes
Copied ../testdir/src/xdiff/xprepare.h to ../tocopyF/xprepare.h, 1058 bytes
Copied ../testdir/src/xdiff/xinclude.h to ../tocopyF/xinclude.h, 1641 bytes
Copied ../testdir/src/xdiff/README.txt to ../tocopyF/README.txt, 647 bytes
Copied ../testdir/src/xdiff/xpatience.c to ../tocopyF/xpatience.c, 10920 bytes
Copied ../testdir/src/xdiff/xdiff.h to ../tocopyF/xdiff.h, 3790 bytes
Copied ../testdir/src/xdiff/xdiffi.h to ../tocopyF/xdiffi.h, 1936 bytes
Copied ../testdir/src/xdiff/xprepare.c to ../tocopyF/xprepare.c, 12242 bytes
Copied ../testdir/src/xdiff/COPYING to ../tocopyF/COPYING, 26421 bytes
Copied ../testdir/src/xdiff/xdiffi.c to ../tocopyF/xdiffi.c, 27698 bytes

---------------STATISTICS--------------------
Consumers: 8 - Buffer Size: 5
Number of Regular Files: 16
Number of FIFO Files: 0
Number of Directories: 0
Number of Other Files: 0
TOTAL BYTES COPIED: 119805
TOTAL TIME: 00:00.111 (min:sec.millis)
==10956==
==10956== HEAP SUMMARY:
==10956==     in use at exit: 0 bytes in 0 blocks
==10956==   total heap usage: 13 allocs, 13 frees, 46,632 bytes allocated
==10956==
==10956== All heap blocks were freed -- no leaks are possible
==10956==
==10956== For lists of detected and suppressed errors, rerun with: -s
==10956== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
mrguven@mrguven:~/Desktop/cse344/hw5/hw5test/put_your_codes_here$
```

Copying folders CTRL+C (Printing)