CSE 344 SYSTEM PROGRAMMING

HOMEWORK #5 REPORT

BUKET GENÇER

210104004298

31.05.2024

Table of Contents

Introduction

This report describes the implementation and experimentation of a directory copying utility called "MWCp" that copies files and subdirectories in parallel using a worker-manager approach to synchronize thread activity. The implementation uses POSIX and Standard C libraries to achieve the task.

Code Structure of Program and Pseudo Code

Code Structure

Description of the functions used:

-main (): Parses command line arguments, sets up signal handling, initializes threads, and prints statistics.

- handle_signal () : Handles SIGINT signal to clean up and exit.

- manager_thread () : Manages the directory copying process and signals completion.

- worker_thread () : Copies files as tasks are added to the buffer.

- copy_directory () : Recursively copies directories and adds file copy tasks to the buffer.

- copy_file () : Performs the actual file copy.

- cleanup_file_task () : Cleans up resources associated with a file task.

- remove_directory () : Removes a directory and its contents.

- print_statistics () : Prints the statistics of the copying process.

Pseudocode

1. **Initialize Program**
    - Parse command line arguments.
    - Set up signal handler for SIGINT.
    - Allocate memory for source and destination directories.
    - Clear destination directory.
    - Initialize statistics variables.

- Record start time.

2. **Create Threads**
    - Create manager thread.
    - Create worker threads.
3. **Manager Thread Function**
    - **manager_thread**:
        - Copy directories recursively from source to destination.
        - Lock buffer mutex, set **done** flag, signal worker threads, unlock buffer mutex.
4. **Worker Thread Function**
    - **worker_thread**:
        - Loop:
            - Lock buffer mutex.
            - Wait if buffer is empty and not done.
            - Exit if buffer is empty and done.
            - Get task from buffer, signal buffer not full, unlock buffer mutex.
            - Copy file and clean up task.
5. **Copy Directory Function**
    - **copy_directory**:
        - Open source directory.
        - For each entry:
            - Skip **.** and **..**.
            - Construct source and destination paths.
            - If directory:
                - Create destination directory, increment directory count, copy recursively.
            - Else:

            - Lock buffer mutex, wait if buffer full, create task, open files.
            - On failure: print error, clean up, unlock mutex, continue.
            - Add task to buffer, increment file count, signal buffer not empty, unlock mutex.
    - Close source directory.

6. **Copy File Function**
    - **copy_file**:
        - Read from source and write to destination in chunks.
        - Update total bytes copied.
        - Handle read/write errors.
        - Close files.
7. **Cleanup File Task Function**
    - **cleanup_file_task**:
        - Free memory for file names.
8. **Remove Directory Function**
    - **remove_directory**:
        - Open directory.
        - For each entry:
            - Skip **.** and **..**.
            - Construct full path.
            - If directory, remove recursively; else, remove file.
        - Close and remove directory.
9. **Print Statistics Function**
    - **print_statistics**:
        - Calculate and print elapsed time, number of files, directories, total bytes copied.
10. **Main Function**

- Parse arguments, set up signal handling, allocate memory, clear destination directory.
- Record start time.
- Create manager and worker threads, wait for completion.
- Record end time, print statistics, free memory, exit.

New Features (Differences between hw4)

The aim of this homework was to make the directory copying tool "MWCp" better by using condition variables and barriers. These changes help manage the buffer better and make sure the worker threads work together properly.

1. **Condition Variables**

Condition variables were added to control the buffer's state. The "buffer not empty" condition variable lets worker threads wait until there are tasks in the buffer. The "buffer not full" condition variable makes the manager thread wait until there is space in the buffer.

2. **Barriers**

Barriers were used to make sure all threads start together. This makes sure that all worker threads are ready before they start copying files.

Initialization of Barriers:

```
pthread_barrier_t barrier;
```

Initialization and Destruction of Barriers in Main:

```
pthread_barrier_init(&barrier, NULL, num_workers + 1); // Initialize barrier with number of workers + 1

pthread_create(&manager, NULL, manager_thread, NULL); // Create manager thread
for (int i = 0; i < num_workers; i++) { // Create worker threads
    pthread_create(&workers[i], NULL, worker_thread, NULL);
}

pthread_barrier_wait(&barrier); // Initial barrier wait to ensure all threads are ready
pthread_barrier_destroy(&barrier); // Destroy barrier after use
```

Worker Thread Synchronization:

```
pthread_barrier_wait(&barrier); // Initial barrier wait to ensure all threads are ready
```

Makefile

The Makefile compiles the C code and provides targets for running the program and conducting tests.

hw5test > put_your_codes_here > Makefile

```makefile
# Makefile for MWCp project

# Compiler
CC = gcc

# Compiler flags
CFLAGS = -Wall -pthread -g

# Source files
SRCS = 210104004298_main.c

# Executable name
EXEC = 210104004298_main

# Default target
all: $(EXEC)

# Build the executable
$(EXEC): $(SRCS)
	$(CC) $(CFLAGS) -o $(EXEC) $(SRCS)

# Clean up the build files
clean:
	rm -f $(EXEC) *.o
	rm -rf ../tocopy/*

# Run the program with default parameters
run: $(EXEC)
	./$(EXEC) 10 4 ../testdir ../tocopy

# Run the tests
test1: clean $(EXEC)
	valgrind --leak-check=full ./$(EXEC) 10 10 ../testdir/src/libvterm ../tocopy

test2: clean $(EXEC)
	./$(EXEC) 10 4 ../testdir/src/libvterm/src ../tocopy

test3: clean $(EXEC)
	./$(EXEC) 10 100 ../testdir ../tocopy

.PHONY: all clean run test1 test2 test3
```

- all: Compiles the code.

- clean : Cleans up build files and the destination directory.

- run : Runs the program with default parameters.

- test1, test2, test3 : Runs the program with different parameters for testing.

Experimentation

If there is no tocopy directory, program create a directory and copy the files.

Test 1

Test1: valgrind ./MWCp 10 10 ../testdir/src/libvterm ../tocopy, #memory leak checking, buffer size=10, number of workers=10, sourceFile, destinationFile

Before Test 1:

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/put_your_codes_here$ cd ..
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test$ ls
put_your_codes_here  testdir  tocopy
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test$ cd tocopy/
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/tocopy$ ls
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/tocopy$
```

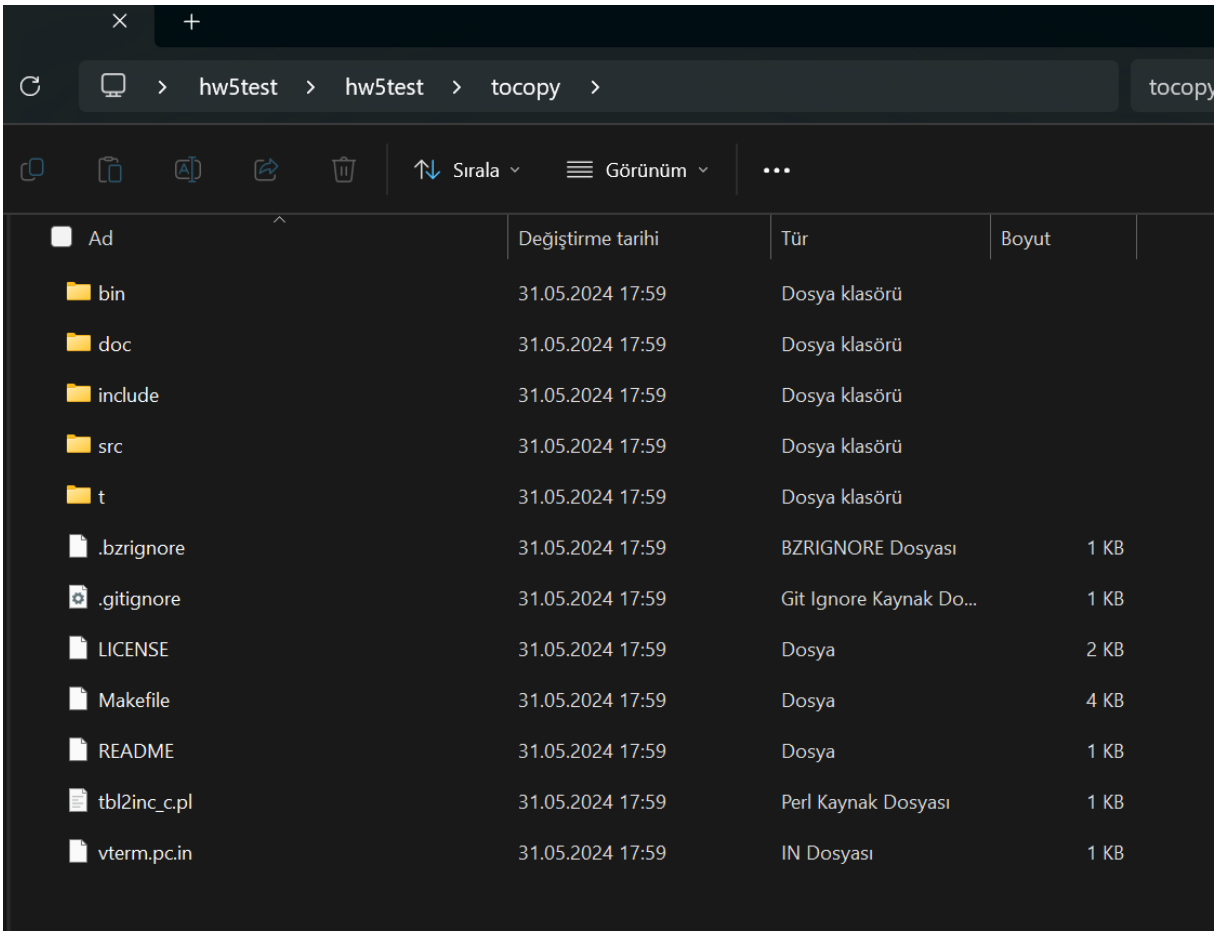
Before test 1 . “tocopy” directory is empty.

After Test 1:

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/put_your_codes_here$ make test1
rm -f 210104004298_main *.o
rm -rf ../tocopy/*
gcc -Wall -pthread -g -o 210104004298_main 210104004298_main.c
valgrind --leak-check=full ./210104004298_main 10 10 ../testdir/src/libvterm ../tocopy
==10201== Memcheck, a memory error detector
==10201== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==10201== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10201== Command: ./210104004298_main 10 10 ../testdir/src/libvterm ../tocopy
==10201==

----------------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: 3.421 seconds
------------------------------------------
==10201==
==10201== HEAP SUMMARY:
==10201==     in use at exit: 0 bytes in 0 blocks
==10201==   total heap usage: 411 allocs, 411 frees, 312,656 bytes allocated
==10201==
==10201== All heap blocks were freed -- no leaks are possible
==10201==
==10201== For lists of detected and suppressed errors, rerun with: -s
==10201== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/put_your_codes_here$ cd ..
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test$ cd tocopy/
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/tocopy$ ls
LICENSE  Makefile  README  bin  doc  include  src  t  tbl2inc_c.pl  vterm.pc.in
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/tocopy$
```



Test 2

Test2: ./MWCp 10 4 ../testdir/src/libvterm/src ../toCopy #buffer size=10, number of workers=4, sourceFile, destinationFile

Before Test2:

Tocopy directory is empty again.

After Test2:

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/put_your_codes_here$ make test2
rm -f 210104004298_main *.o
rm -rf ../tocopy/*
gcc -Wall -pthread -g -o 210104004298_main 210104004298_main.c
./210104004298_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: 1.769 seconds
------------------------------------------
```

Test 3

Test3: ./MWCp 10 10 ../testdir ../toCopy #buffer size=10, number of workers=10, sourceFile, destinationFile

Before Test 3:

Tocopy directory is empty again.

After Test 3:

```
----------------------------------------
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/hw5test/hw5test/put_your_codes_here$ make test3
rm -f 210104004298_main *.o
rm -rf ../tocopy/*
gcc -Wall -pthread -g -o 210104004298_main 210104004298_main.c
./210104004298_main 10 100 ../testdir ../tocopy

----------------STATISTICS----------------
Consumers: 100 - Buffer Size: 10
Number of Regular Files: 3116
Number of FIFO Files: 0
Number of Directories: 151
TOTAL BYTES COPIED: 73520554
TOTAL TIME: 37.515 seconds
----------------------------------------
```

Handling Invalid Commands

Some screenshoots of invalid commands examples:

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/system_hw4/hw4test/hw4test/put_your_codes_here$ ./MWCp 0 0 ../testdir ../toc
opy
Buffer size and number of workers must be greater than 0
```

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/system_hw4/hw4test/hw4test/put_your_codes_here$ ./MWCp 1  a b 0 0 ../testdir
 ../tocopy
Usage: ./MWCp <buffer_size> <num_workers> <src_dir> <dest_dir>
```

```
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/system_hw4/hw4test/hw4test/put_your_codes_here$ ./MWCp 10 10 ../nofileeee ../tocopy
Failed to open source directory: No such file or directory

----------------STATISTICS----------------
Consumers: 10 - Buffer Size: 10
Number of Regular Files: 0
Number of FIFO Files: 0
Number of Directories: 0
TOTAL BYTES COPIED: 0
TOTAL TIME: 0.002 seconds
----------------------------------------
```

Handling SIGINT (Ctrl+C)

```
----------------------------------------
bktgncr@DESKTOP-AI758A5:/mnt/c/Users/HUAWEI/OneDrive/Masaüstü/system_hw4/hw4test/hw4test/put_your_codes_here$ ./MWCp 10 10 ../testdir ../tocopy
^C
Process interrupted. Cleaning up and exiting...
```