

CSE344 – System Programming - HW4 REPORT

Reşit Aydın - 200104004019

1. General Structure & Logic

1.1 Worker Threads Logic:

Worker threads continuously retrieve files from a shared buffer, copy their contents from source to destination, update counters for files processed and bytes copied, and terminate upon receiving termination signals from the manager or user interrupt.

```
void *worker_thread(void *arg) {
    while (1) {
        pthread_mutex_lock(&file_buffer.mutex);

        while (file_buffer.count == 0 && !file_buffer.done && !stop) {
            pthread_cond_wait(&file_buffer.not_empty, &file_buffer.mutex);
        }

        if (file_buffer.count == 0 && (file_buffer.done || stop)) {
            pthread_mutex_unlock(&file_buffer.mutex);
            break;
        }

        file_info_t file_info = file_buffer.buffer[file_buffer.out];
        file_buffer.out = (file_buffer.out + 1) % buffer_size;
        file_buffer.count--;

        pthread_cond_signal(&file_buffer.not_full);
        pthread_mutex_unlock(&file_buffer.mutex);

        char buffer[4096];
        ssize_t bytes_read, bytes_written;
        size_t total_bytes = 0;
        while ((bytes_read = read(file_info.src_fd, buffer, sizeof(buffer))) > 0) {
            bytes_written = write(file_info.dst_fd, buffer, bytes_read);
            if (bytes_written != bytes_read) {
                perror("write");
                break;
            }
            total_bytes += bytes_written;
        }

        close(file_info.src_fd);
        close(file_info.dst_fd);

        pthread_mutex_lock(&file_buffer.mutex);
        files_copied++;
        total_bytes_copied += total_bytes;
        pthread_mutex_unlock(&file_buffer.mutex);
    }

    return NULL;
}
```

1.2 Manager Thread Logic

The manager thread coordinates the initiation of directory processing, sets flags to signal termination to worker threads, and oversees the completion of directory processing.

```
void *manager_thread(void *arg) {
    char **dirs = (char **)arg;
    char *src_dir = dirs[0];
    char *dst_dir = dirs[1];

    process_directory(src_dir, dst_dir);

    pthread_mutex_lock(&file_buffer.mutex);
    file_buffer.done = 1;
    pthread_cond_broadcast(&file_buffer.not_empty);
    pthread_mutex_unlock(&file_buffer.mutex);

    return NULL;
}
```

1.3 Ctrl^C Signal Handling Logic

The handle_signal function sets a flag to stop processing when SIGINT is received. Then it is handled using sigaction in main function.

```
void handle_signal(int signal) {
    stop = 1;
}
```

```
// Handling Ctrl^C signal using sigaction
struct sigaction sa;
sa.sa_handler = handle_signal;
sa.sa_flags = 0;
sigemptyset(&sa.sa_mask);
sigaction(SIGINT, &sa, NULL);
```

```
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$ ./MWCP 10 10 ../testdir ../toCopy
^C
-----STATISTICS-----
Workers: 10 - Buffer Size: 10
Number of Regular Files: 1734
Number of FIFOs: 0
Number of Directories: 65
Number of Symbolic Links: 0
TOTAL BYTES COPIED: 60694721
TOTAL TIME(min:sec.mili): 0:00.181
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$
```

When Ctrl ^ C is pressed during the runtime of the program, the program displays the current statistics and exists right away.

1.4 Main Function Structure

The main function of the program initializes various variables and data structures, sets up signal handling for Ctrl^C, creates threads for the manager and worker functions, waits for them to finish their tasks, calculates the elapsed time, destroys mutex and condition variables, and finally prints out the statistics gathered during execution.

2. Compiling & Running

- The program can be compiled using the following command:

make

- The program can be executed like this (ex.):

./MWCp 10 10 ../testdir ../toCopy

#buffer size=10, number of workers=10, sourceFile, destinationFile

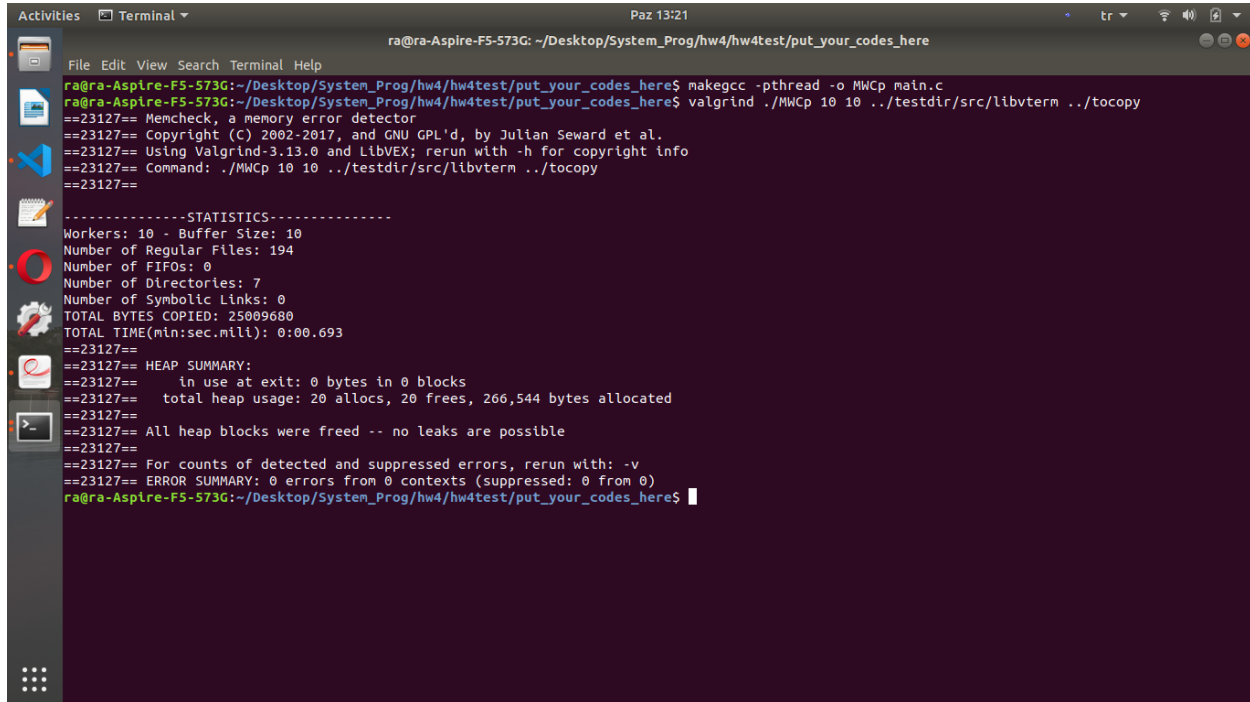
- The generated output files can be cleaned using this command:

make clean

3. Test Cases and Results

These test cases are the ones given in the pdf. I put the code into the directory called “put_your_codes_here” then run it.

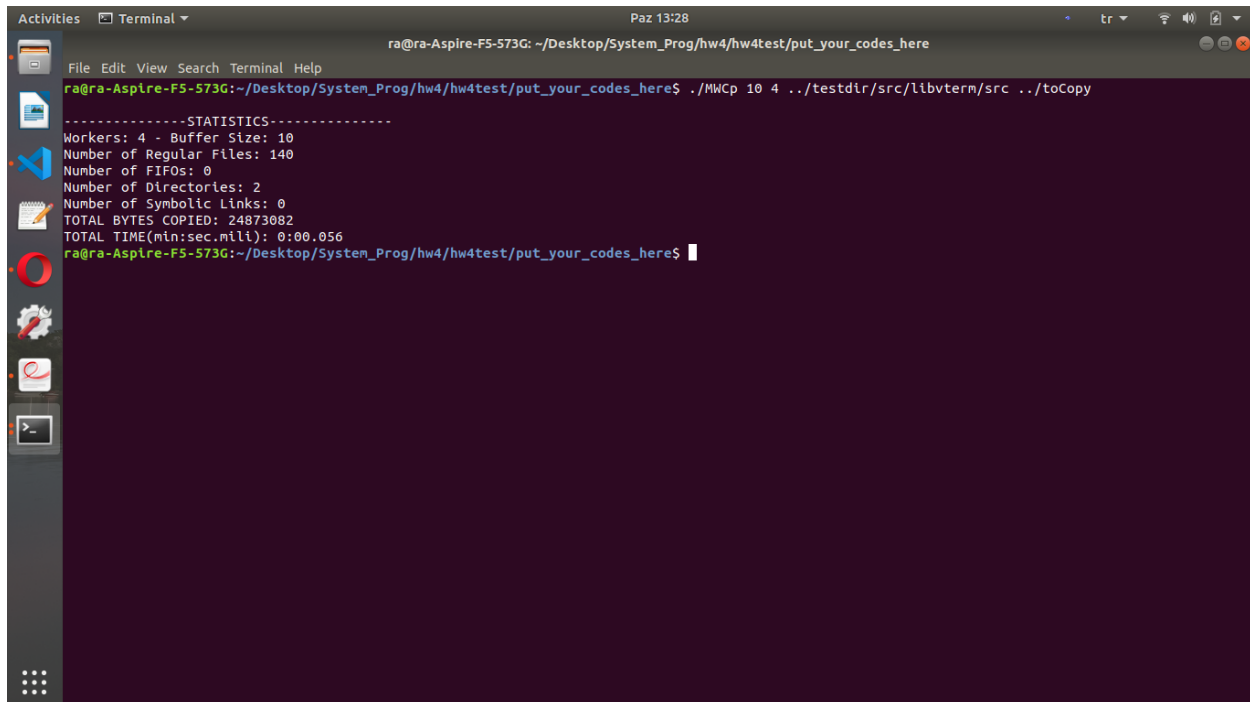
3.1 Test 1 Output



```
ra@ra-Aspire-F5-573G: ~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$ makegcc -pthread -o MMCP main.c
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$ valgrind ./MMCP 10 10 ../testdir/src/libvterm ../tocopy
==23127== Memcheck, a memory error detector
==23127== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==23127== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==23127== Command: ./MMCP 10 10 ../testdir/src/libvterm ../tocopy
==23127==

-----STATISTICS-----
Workers: 10 - Buffer Size: 10
Number of Regular Files: 194
Number of FIFOs: 0
Number of Directories: 7
Number of Symbolic Links: 0
TOTAL BYTES COPIED: 25009680
TOTAL TIME(min:sec.mill): 0:00.693
==23127==
==23127== HEAP SUMMARY:
==23127==   in use at exit: 0 bytes in 0 blocks
==23127==   total heap usage: 20 allocs, 20 frees, 266,544 bytes allocated
==23127==
==23127== All heap blocks were freed -- no leaks are possible
==23127==
==23127== For counts of detected and suppressed errors, rerun with: -v
==23127== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$
```

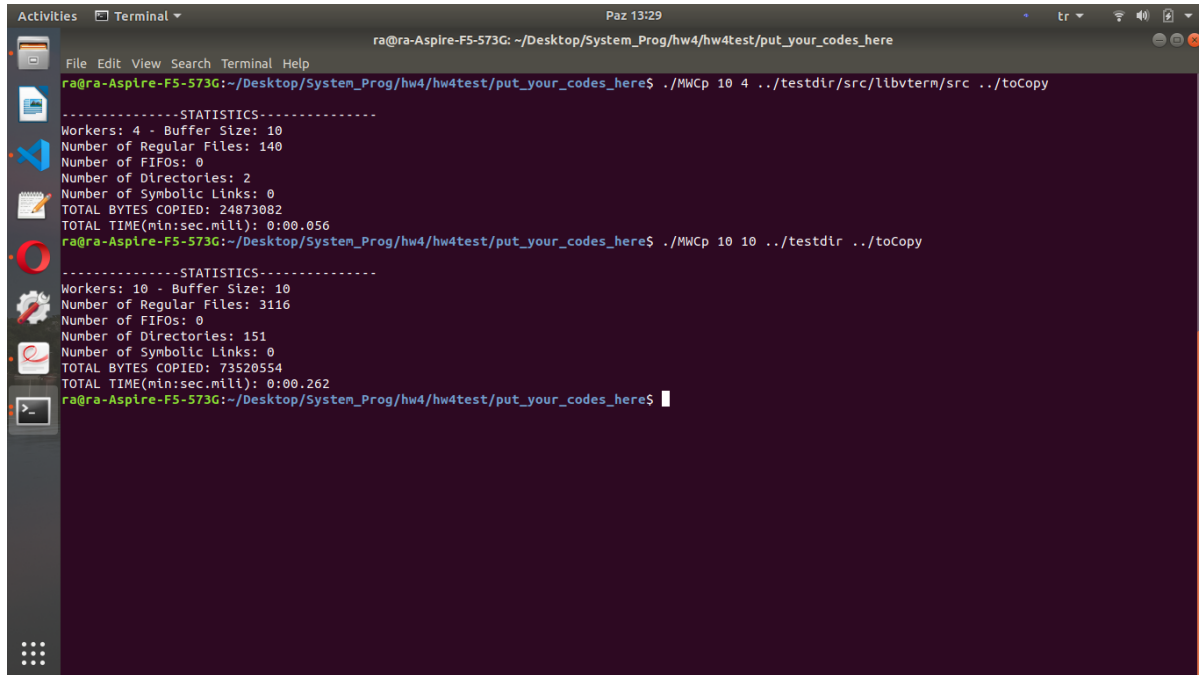
3.2 Test 2 Output



```
ra@ra-Aspire-F5-573G: ~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$ ./MMCP 10 4 ../testdir/src/libvterm/src ../toCopy

-----STATISTICS-----
Workers: 4 - Buffer Size: 10
Number of Regular Files: 140
Number of FIFOs: 0
Number of Directories: 2
Number of Symbolic Links: 0
TOTAL BYTES COPIED: 24873082
TOTAL TIME(min:sec.mill): 0:00.056
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$
```

3.3 Test 3 Output



```
ra@ra-Aspire-F5-573G: ~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$ ./MMCP 10 4 ../testdir/src/libvterm/src ../toCopy
-----STATISTICS-----
Workers: 4 - Buffer Size: 10
Number of Regular Files: 140
Number of FIFOs: 0
Number of Directories: 2
Number of Symbolic Links: 0
TOTAL BYTES COPIED: 24873082
TOTAL TIME(min:sec.mlll): 0:00.056
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$ ./MMCP 10 10 ../testdir ../toCopy
-----STATISTICS-----
Workers: 10 - Buffer Size: 10
Number of Regular Files: 3116
Number of FIFOs: 0
Number of Directories: 151
Number of Symbolic Links: 0
TOTAL BYTES COPIED: 73520554
TOTAL TIME(min:sec.mlll): 0:00.262
ra@ra-Aspire-F5-573G:~/Desktop/System_Prog/hw4/hw4test/put_your_codes_here$
```

4. Conclusion

My program works as expected. As I tested it with valgrind, there are not any memory leaks. Also, using thread synchronization techniques like mutexes and condition variables, there are no race conditions or any other problems.