Alper Tavşanoğlu-210104004142-HW-4-Report

The purpose of this program is to perform parallel file and directory copying using Threads and POSIX synchronization fundamentals. Program is designed to handle these tasks: copying files and directories to new directory, each managed by designated worker threads pool in a synchronized computing environment. This system controls file and directory operations through thread synchronization, ensuring that each type of operation is handled by the appropriate worker thread. The program is designed to run until all specified files and directories have been successfully copied.

System Design

Main Program

The program, accepts buffer size, number of workers, and source/destination directory paths as input arguments. Checks for the correct number of arguments and print a usage message if they are incorrect.

Initializes and manages worker threads, ensuring efficient task distribution and synchronization. All mutexes and condition variables are destroyed correctly. Condition variables used to manage worker sleep/wake cycles is appropriate and avoids busy waiting.

Tracks and reports the time taken to copy the entire directory content, along with detailed statistics about the types and numbers of files copied. Records and outputs detailed statistics about the operation, which is a key requirement.

Signal handling is implemented to allow the program to terminate gracefully in response to interruptions like SIGINT or SIGTERM. A custom signal handler sets a termination flag, which is checked by threads to conclude their operations safely.

Manager

In the program, The manager is implemented within the main function.It Reads from the source directory and prepares tasks for workers by creating appropriate destination files, handling truncation if files already exist.

Provides robust error reporting if file operations fail and manages file descriptors securely.

Controls a shared buffer, ensuring that the workers can access tasks as they are ready without conflicts or resource contention. The SharedBuffer is central to synchronization, equipped with mutexes and condition variables. It stores file tasks, tracks the number of tasks, and signals task availability to workers. This buffer is critical for managing the flow of data between the manager and workers and ensuring that no two workers process the same file simultaneously.

Signals completion of task preparation to allow graceful termination of worker threads.

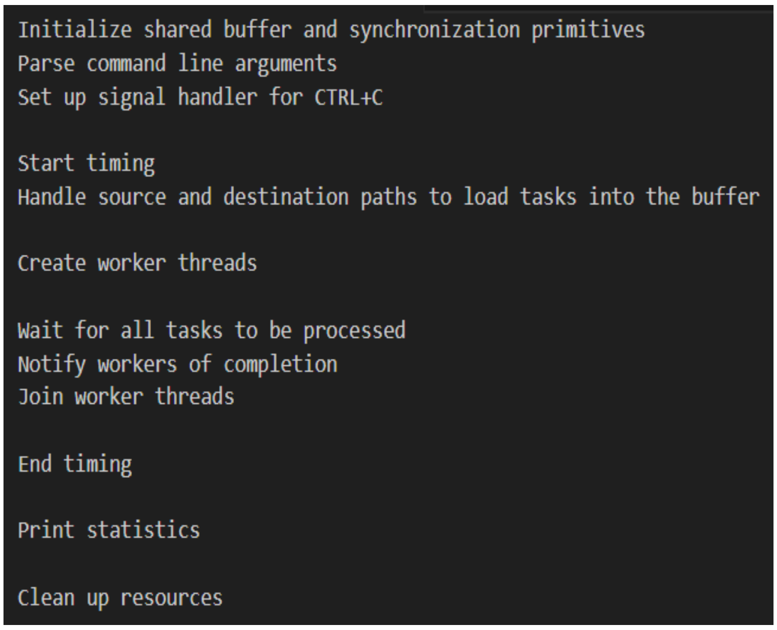
Worker

Workers pull tasks from the shared buffer, handling file copying from the source to the destination. This involves opening source files, reading content, and writing to destination files while managing file-specific actions such as truncation and error handling.

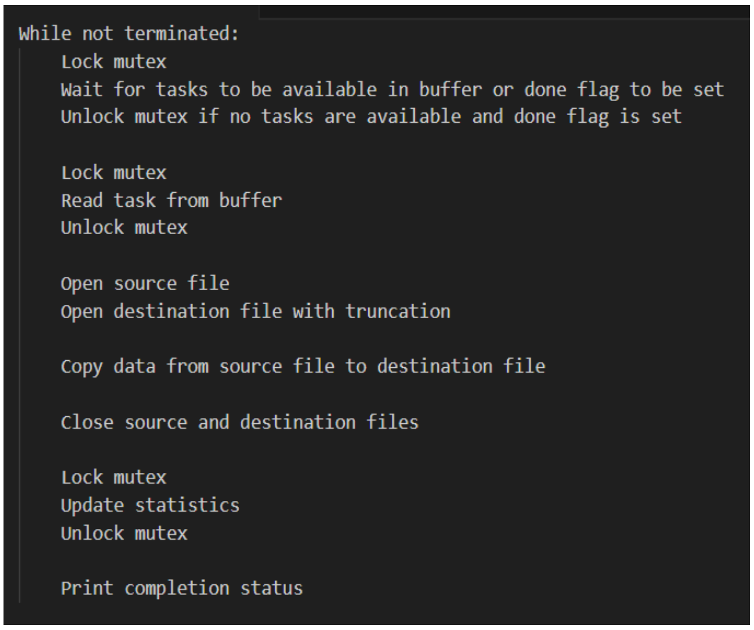
Each worker is responsible for managing its file descriptors and allocated resources, ensuring that all resources are freed and closed appropriately after task completion.

Workers utilize mutexes and condition variables from the shared buffer to synchronize their access to the buffer, ensuring efficient and conflict-free task processing. They also handle their termination based on the shared termination flag set by the signal handler.

Basic Pseudocodes



```
Initialize shared buffer and synchronization primitives
Parse command line arguments
Set up signal handler for CTRL+C

Start timing
Handle source and destination paths to load tasks into the buffer

Create worker threads

Wait for all tasks to be processed
Notify workers of completion
Join worker threads

End timing

Print statistics

Clean up resources
```



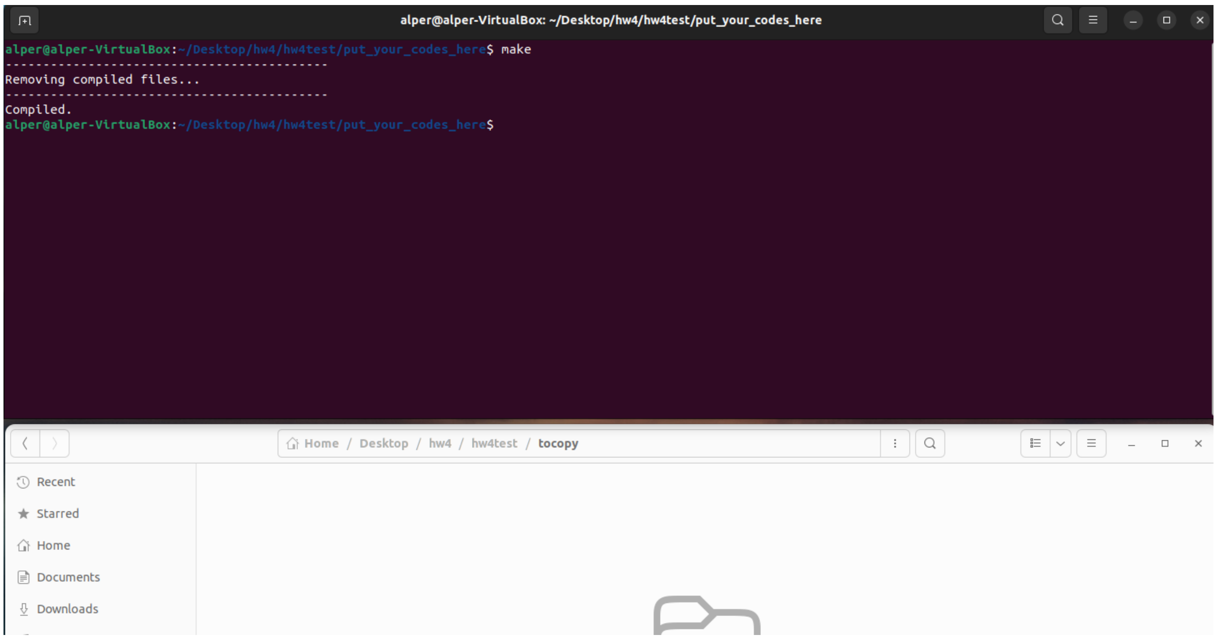
```
While not terminated:
    Lock mutex
    Wait for tasks to be available in buffer or done flag to be set
    Unlock mutex if no tasks are available and done flag is set

    Lock mutex
    Read task from buffer
    Unlock mutex

    Open source file
    Open destination file with truncation

    Copy data from source file to destination file

    Close source and destination files

    Lock mutex
    Update statistics
    Unlock mutex

    Print completion status
```

Compile and Tests

For compile the code we can use Makefile. With **make** command program will compiled. Makfile also include **makeclean** command.

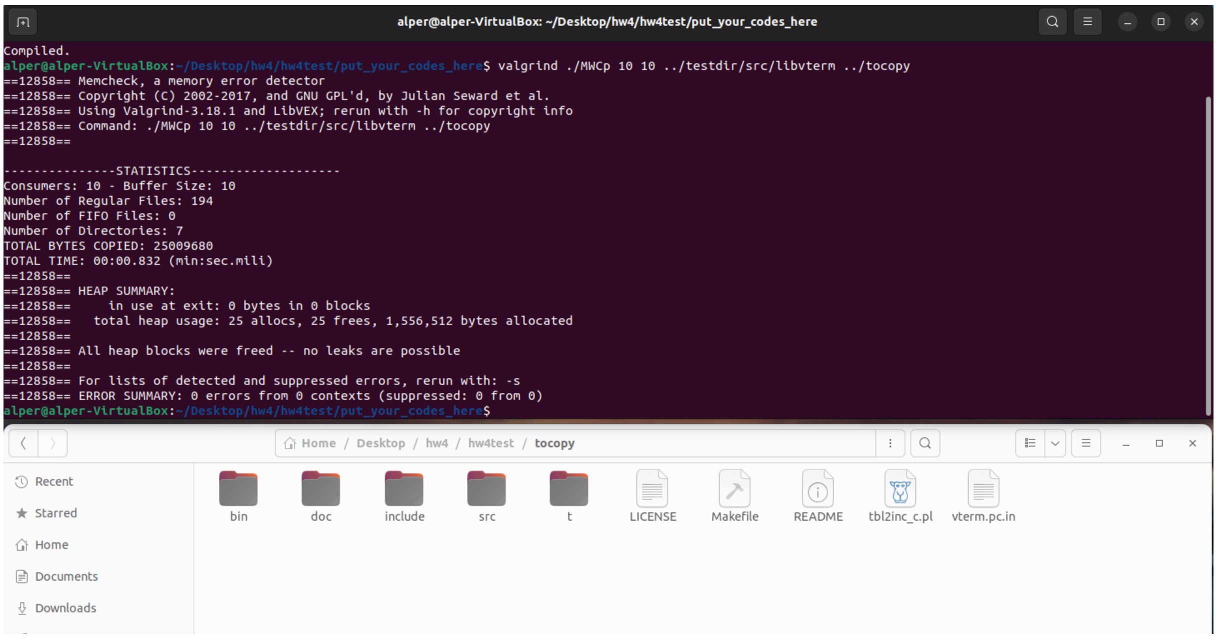
Test1: valgrind ./MWCp 10 10 ../testdir/src/libvterm ../tocopy, #memory leak checking, buffer size=10, number of workers=10, sourceFile, destinationFile

First Compile and Before Run Test1


alper@alper-VirtualBox: ~/Desktop/hw4/hw4test/put_your_codes_here
Home / Desktop / hw4 / hw4test / tocopy
Recent
Starred
Home
Documents
Downloads

```
alper@alper-VirtualBox:~/Desktop/hw4/hw4test/put_your_codes_here$ make
--------------------------------------------
Removing compiled files...
--------------------------------------------
Compiled.
alper@alper-VirtualBox:~/Desktop/hw4/hw4test/put_your_codes_here$
```

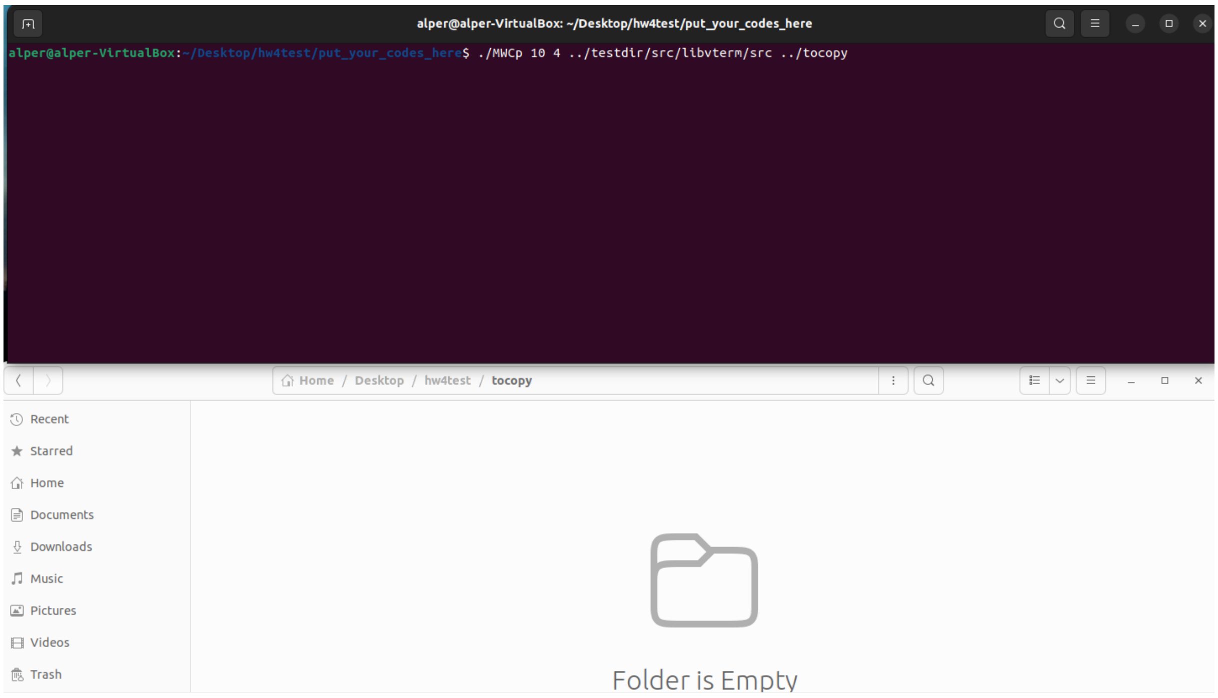
Then Run Test1


alper@alper-VirtualBox: ~/Desktop/hw4/hw4test/put_your_codes_here
Home / Desktop / hw4 / hw4test / tocopy
Recent
Starred
Home
Documents
Downloads
bin
doc
include
src
t
LICENSE
Makefile
README
tbl2inc_c.pl
vterm.pc.in

```
Compiled.
alper@alper-VirtualBox:~/Desktop/hw4/hw4test/put_your_codes_here$ valgrind ./MWCp 10 10 ../testdir/src/libvterm ../tocopy
==12858== Memcheck, a memory error detector
==12858== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==12858== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==12858== Command: ./MWCp 10 10 ../testdir/src/libvterm ../tocopy
==12858==

--------------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: 00:00.832 (min:sec.mili)
==12858==
==12858== HEAP SUMMARY:
==12858==     in use at exit: 0 bytes in 0 blocks
==12858==   total heap usage: 25 allocs, 25 frees, 1,556,512 bytes allocated
==12858==
==12858== All heap blocks were freed -- no leaks are possible
==12858==
==12858== For lists of detected and suppressed errors, rerun with: -s
==12858== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
alper@alper-VirtualBox:~/Desktop/hw4/hw4test/put_your_codes_here$
```

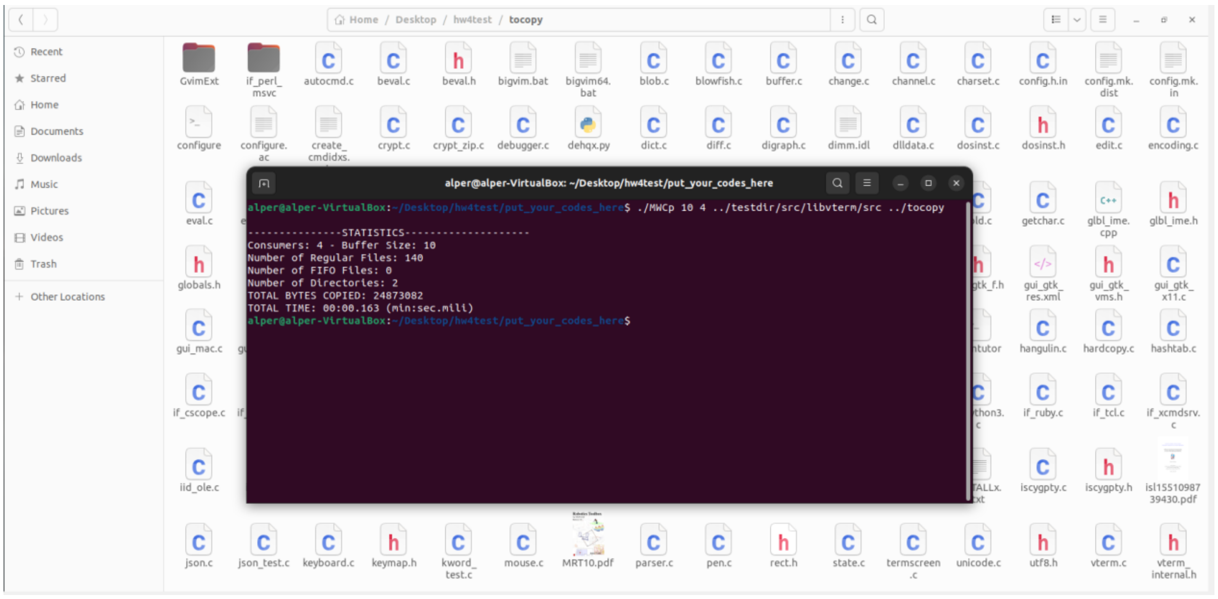
Test2: ./MWCp 10 4 ../testdir/src/ libvterm/src ../tocopy #buffer size=10, number of workers=4, sourceFile, destinationFile

Before Run Test2



```
alper@alper-VirtualBox:~/Desktop/hw4test/put_your_codes_here$ ./MWCp 10 4 ../testdir/src/libvterm/src ../tocopy
```

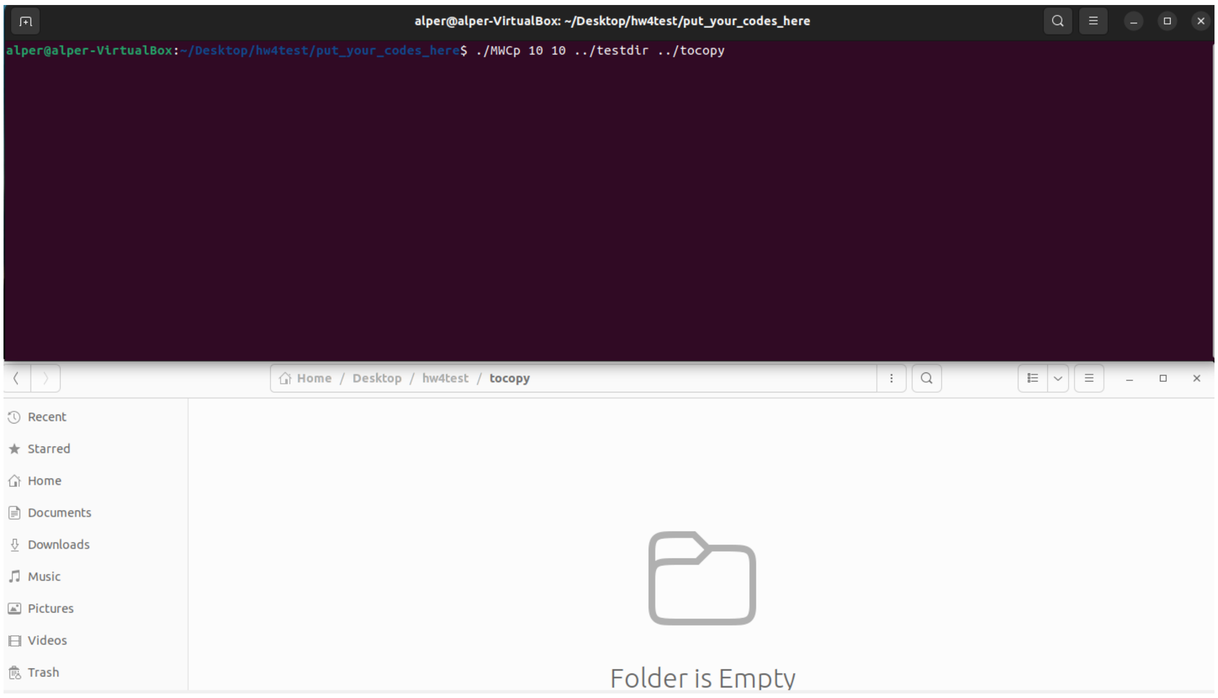
After Run Test2



```
alper@alper-VirtualBox:~/Desktop/hw4test/put_your_codes_here$ ./MWCp 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: 00:00.163 (min:sec.mili)
alper@alper-VirtualBox:~/Desktop/hw4test/put_your_codes_here$
```

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

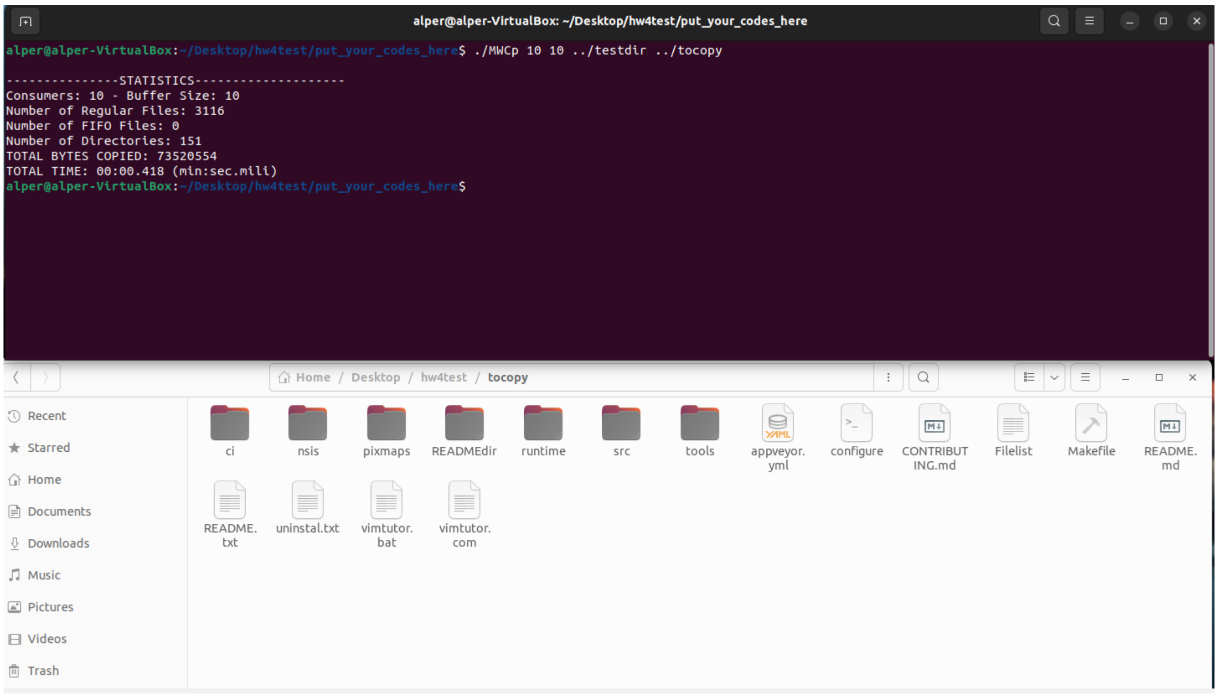
Before Run Test3



```
alper@alper-VirtualBox:~/Desktop/hw4test/put_your_codes_here$ ./MWCp 10 10 ../testdir ../tocopy
```

alper@alper-VirtualBox: ~/Desktop/hw4test/put_your_codes_here
Home / Desktop / hw4test / tocopy
Recent
Starred
Home
Documents
Downloads
Music
Pictures
Videos
Trash
Folder is Empty

Run Test3



```
alper@alper-VirtualBox:~/Desktop/hw4test/put_your_codes_here$ ./MWCp 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: 00:00.418 (min:sec.milli)
alper@alper-VirtualBox:~/Desktop/hw4test/put_your_codes_here$
```