SPRING 2023 – CSE 344 SYSTEM PROGRAMMING

HOMEWORK 5

MERT GÜRŞİMŞİR

1901042646

How to run?

First you should use make command.

Then you are going to have 1 executable files which is “hw5”.

You can use it like follows:

```
./hw5 <SIZE OF BUFFER> <# OF CONSUMER> <SOURCE> <DESTINATION>
```

After that, you can use “make clean” to delete .o file and executable.

System Architecture, Design Decisions & Implementation Details

First of all, I have created 1 producer thread and given number of consumer threads to create thread pool. Producer thread are filling the buffer and consumers are consuming the buffer in the order tasks are added. It is like a queue.

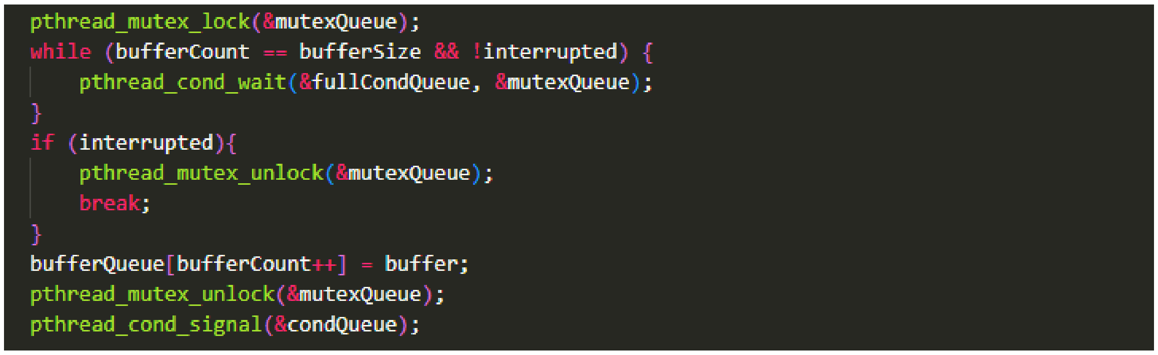
My buffer keeps followings as struct:

- source file descriptor
- destination file descriptor
- file name

Queue of the buffer keeps this structs.

PRODUCER thread is simply adding new structs to the buffer queue. It is iterating through the files in the given source directory. If current item is a regular file, it opens the file for reading mode in source path and for writing mode in destination path.

This code snippet is for critical region:



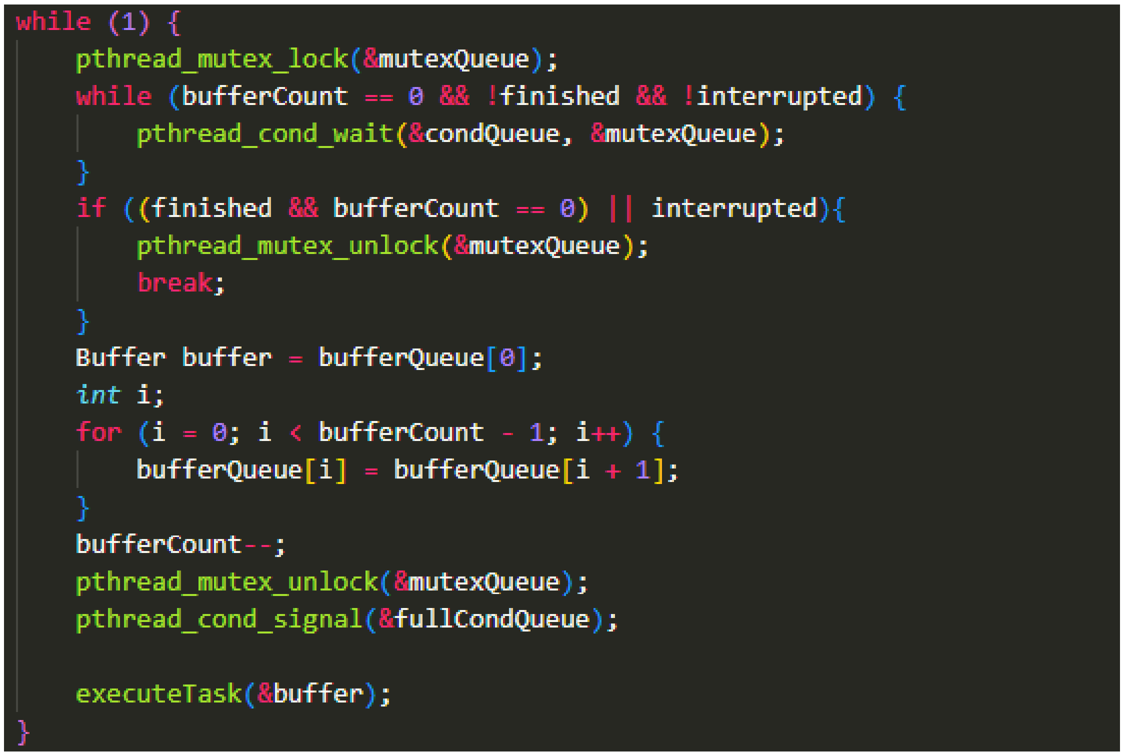
```
pthread_mutex_lock(&mutexQueue);
while (bufferCount == bufferSize && !interrupted) {
    pthread_cond_wait(&fullCondQueue, &mutexQueue);
}
if (interrupted){
    pthread_mutex_unlock(&mutexQueue);
    break;
}
bufferQueue[bufferCount++] = buffer;
pthread_mutex_unlock(&mutexQueue);
pthread_cond_signal(&condQueue);
```

Here, mutexQueue mutex is locked during adding new element to the bufferQueue. If we are exceeding the buffer size, then we are waiting for a consumer to consume the first struct in the array. If interrupt signal is send, then producer finishes its execution.

For the subdirectories, I am going through same procedures recursively as it is stated in PDF.

CONSUMER takes the first element from the buffer queue and copy the related file from source path to destination path.

This code snippet is for consumer and its critical region:



```
while (1) {
    pthread_mutex_lock(&mutexQueue);
    while (bufferCount == 0 && !finished && !interrupted) {
        pthread_cond_wait(&condQueue, &mutexQueue);
    }
    if ((finished && bufferCount == 0) || interrupted){
        pthread_mutex_unlock(&mutexQueue);
        break;
    }
    Buffer buffer = bufferQueue[0];
    int i;
    for (i = 0; i < bufferCount - 1; i++) {
        bufferQueue[i] = bufferQueue[i + 1];
    }
    bufferCount--;
    pthread_mutex_unlock(&mutexQueue);
    pthread_cond_signal(&fullCondQueue);

    executeTask(&buffer);
}
```

Consumer is simply doing these continuously:

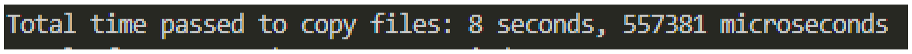
- Take next element from the queue.
- Copy it by executeTask method.

While taking next element from the queue, critical section is created. Of course, if there is no element in the queue and files is not finished and program is not interrupted, then we have to wait here.

Executing task is simply copying the current file.

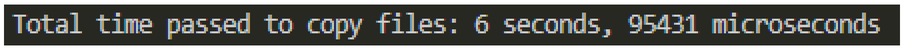
I have experimented with different number of consumer threads and different buffer sizes.

Buffer size = 10, Consumer threads = 10:



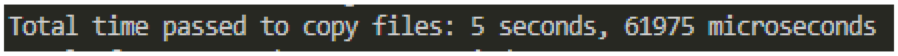
```
Total time passed to copy files: 8 seconds, 557381 microseconds
```

Buffer size = 100, Consumer threads = 10:



```
Total time passed to copy files: 6 seconds, 95431 microseconds
```

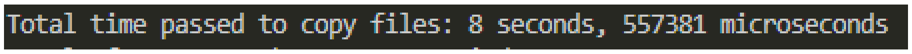
Buffer size = 1000, Consumer threads = 10:



```
Total time passed to copy files: 5 seconds, 61975 microseconds
```

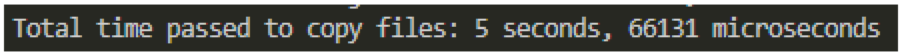
It can be seen that when we increase the buffer size, program becomes faster because producer doesn’t have to wait for consumer to consume an item. It can add to buffer because there is enough space.

Buffer size = 10, Consumer threads = 10:



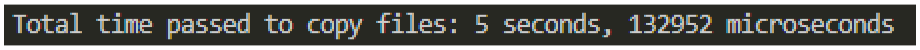
```
Total time passed to copy files: 8 seconds, 557381 microseconds
```

Buffer size = 10, Consumer threads = 100:



```
Total time passed to copy files: 5 seconds, 66131 microseconds
```

Buffer size = 10, Consumer threads = 1000:



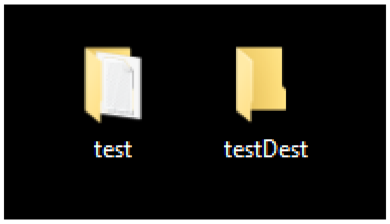
```
Total time passed to copy files: 5 seconds, 132952 microseconds
```

It can be seen that when we increase the number of consumer threads, program becomes faster because consumer threads are copying files in a parallel manner. I think number 1000 is sufficient because program doesn’t get much faster after that number. This may because one file in my directory is big so we are waiting for that file to finish copying.

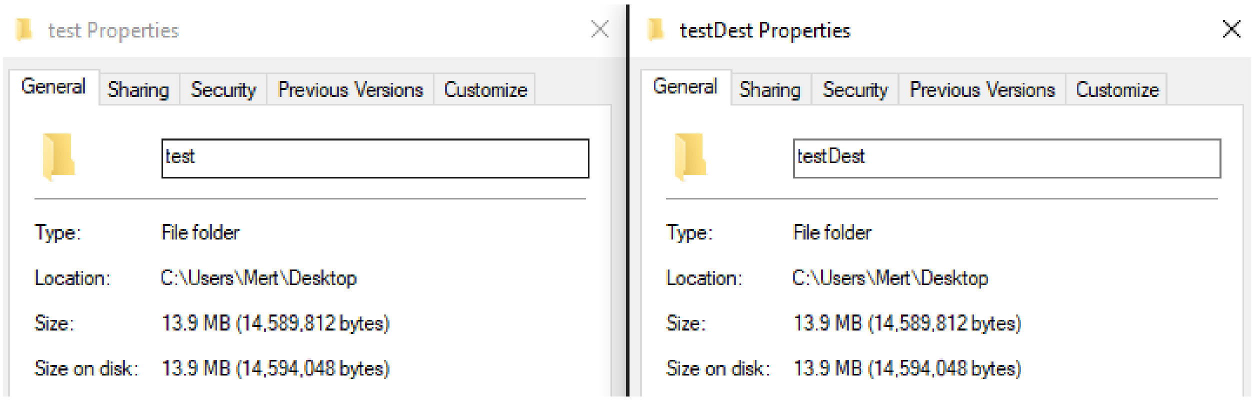
I set the limit for open file descriptors 1020. It is 1024 for Linux processes. I have used semaphores. I am waiting for semaphore when I am opening the file and I am posting it when I am closing the file. So this doesn’t produce an error.

TESTS

Files should be copied correctly (Destination folder already exists):



test folder should be created in testDest.



test Properties

test

Type: File folder

Location: C:\Users\Mert\Desktop

Size: 13.9 MB (14,589,812 bytes)

Size on disk: 13.9 MB (14,594,048 bytes)

testDest Properties

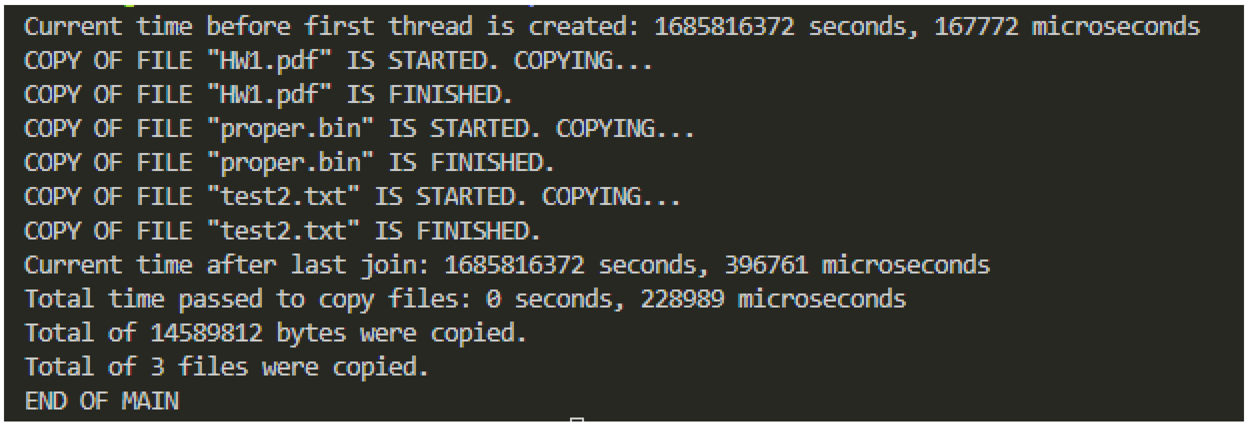
testDest

Type: File folder

Location: C:\Users\Mert\Desktop

Size: 13.9 MB (14,589,812 bytes)

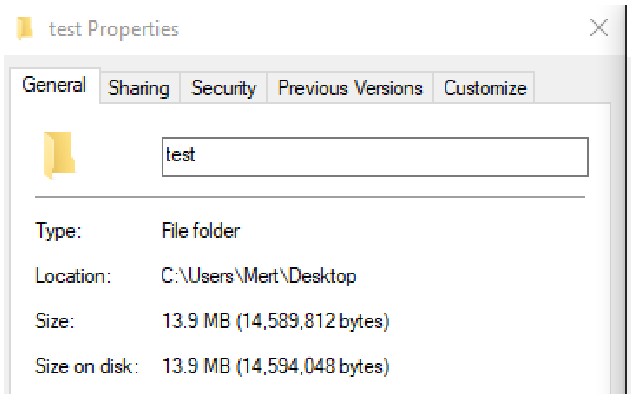
Size on disk: 13.9 MB (14,594,048 bytes)



```
Current time before first thread is created: 1685816372 seconds, 167772 microseconds
COPY OF FILE "HW1.pdf" IS STARTED. COPYING...
COPY OF FILE "HW1.pdf" IS FINISHED.
COPY OF FILE "proper.bin" IS STARTED. COPYING...
COPY OF FILE "proper.bin" IS FINISHED.
COPY OF FILE "test2.txt" IS STARTED. COPYING...
COPY OF FILE "test2.txt" IS FINISHED.
Current time after last join: 1685816372 seconds, 396761 microseconds
Total time passed to copy files: 0 seconds, 228989 microseconds
Total of 14589812 bytes were copied.
Total of 3 files were copied.
END OF MAIN
```

Files should be copied correctly (Destination folder doesn't exist):

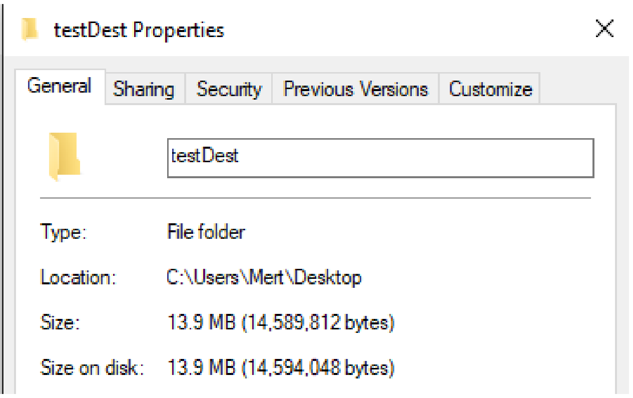
test folder shouldn't be created in testDest, instead all files in test folder should be copied in testDest directly.



test Properties

General | Sharing | Security | Previous Versions | Customize

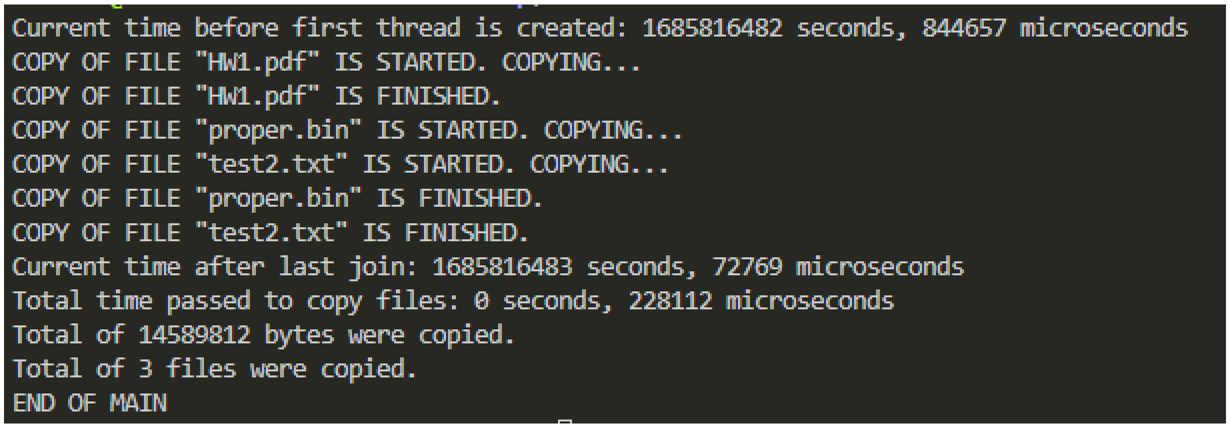
```
test
Type:          File folder
Location:      C:\Users\Mert\Desktop
Size:          13.9 MB (14,589,812 bytes)
Size on disk:  13.9 MB (14,594,048 bytes)
```



testDest Properties

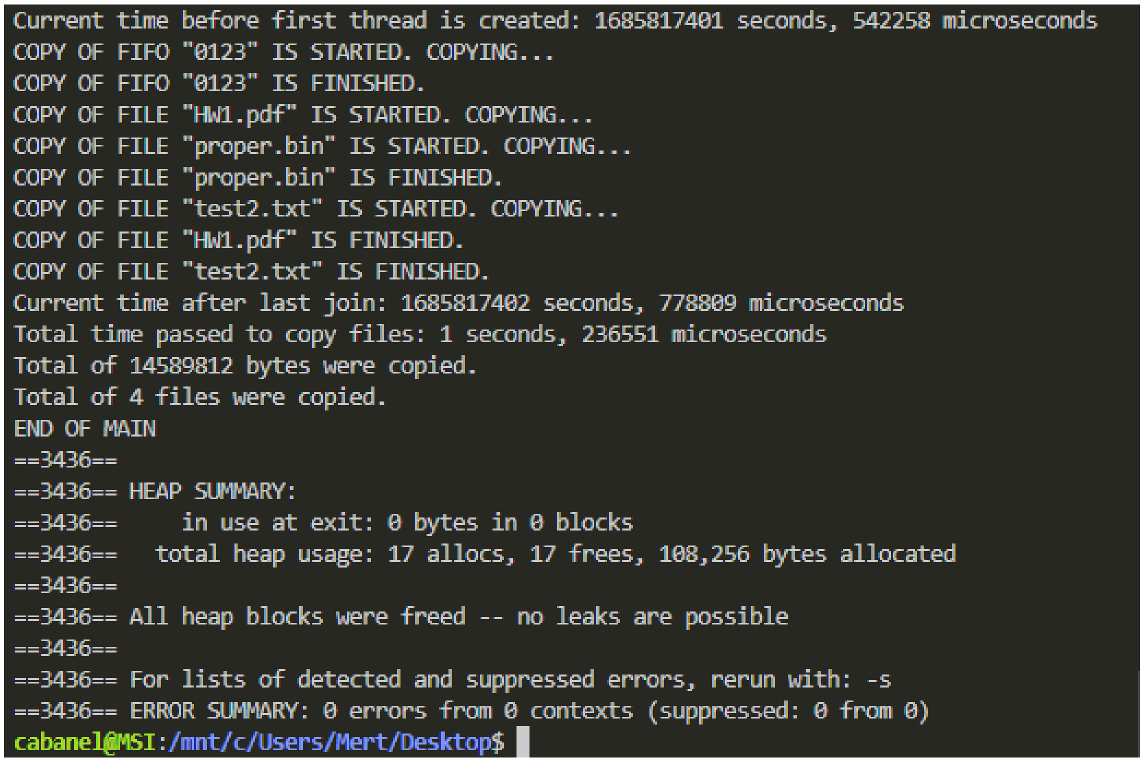
General | Sharing | Security | Previous Versions | Customize

```
testDest
Type:          File folder
Location:      C:\Users\Mert\Desktop
Size:          13.9 MB (14,589,812 bytes)
Size on disk:  13.9 MB (14,594,048 bytes)
```

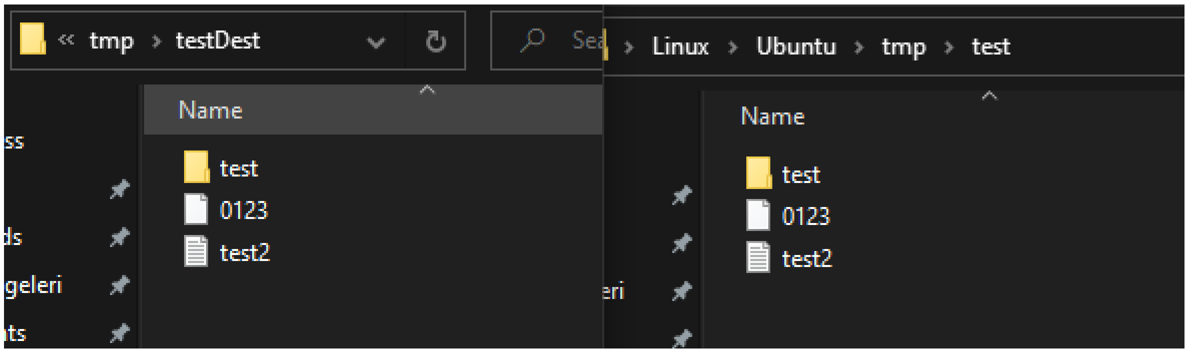


```
Current time before first thread is created: 1685816482 seconds, 844657 microseconds
COPY OF FILE "HW1.pdf" IS STARTED. COPYING...
COPY OF FILE "HW1.pdf" IS FINISHED.
COPY OF FILE "proper.bin" IS STARTED. COPYING...
COPY OF FILE "test2.txt" IS STARTED. COPYING...
COPY OF FILE "proper.bin" IS FINISHED.
COPY OF FILE "test2.txt" IS FINISHED.
Current time after last join: 1685816483 seconds, 72769 microseconds
Total time passed to copy files: 0 seconds, 228112 microseconds
Total of 14589812 bytes were copied.
Total of 3 files were copied.
END OF MAIN
```

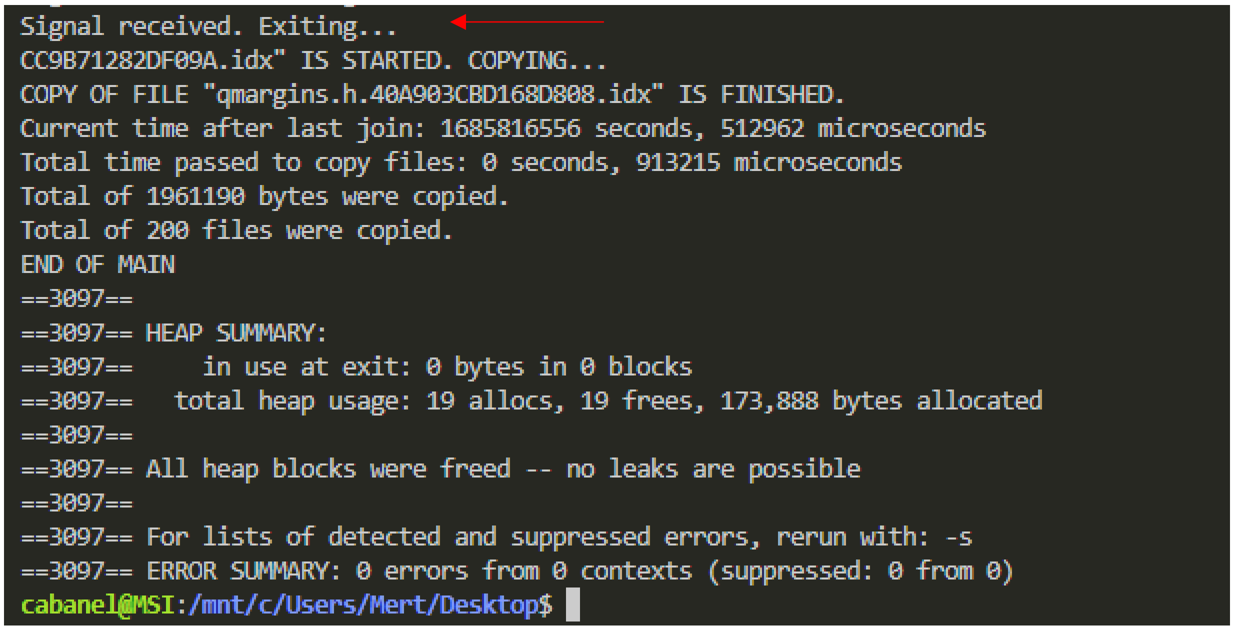
<u>FIFOs should be copied correctly:</u>



```
Current time before first thread is created: 1685817401 seconds, 542258 microseconds
COPY OF FIFO "0123" IS STARTED. COPYING...
COPY OF FIFO "0123" IS FINISHED.
COPY OF FILE "HW1.pdf" IS STARTED. COPYING...
COPY OF FILE "proper.bin" IS STARTED. COPYING...
COPY OF FILE "proper.bin" IS FINISHED.
COPY OF FILE "test2.txt" IS STARTED. COPYING...
COPY OF FILE "HW1.pdf" IS FINISHED.
COPY OF FILE "test2.txt" IS FINISHED.
Current time after last join: 1685817402 seconds, 778809 microseconds
Total time passed to copy files: 1 seconds, 236551 microseconds
Total of 14589812 bytes were copied.
Total of 4 files were copied.
END OF MAIN
==3436==
==3436== HEAP SUMMARY:
==3436==     in use at exit: 0 bytes in 0 blocks
==3436==   total heap usage: 17 allocs, 17 frees, 108,256 bytes allocated
==3436==
==3436== All heap blocks were freed -- no leaks are possible
==3436==
==3436== For lists of detected and suppressed errors, rerun with: -s
==3436== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cabanel@MSI:/mnt/c/Users/Mert/Desktop$
```

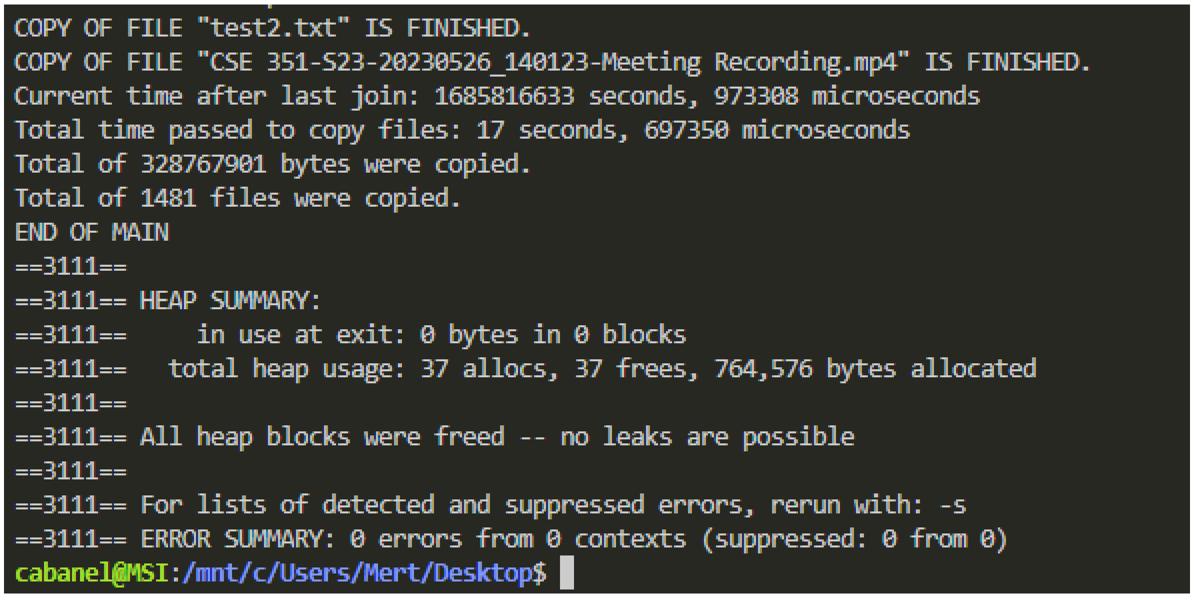


Signals should be handled properly and program should terminate:



```
Signal received. Exiting...
CC9B71282DF09A.idx" IS STARTED. COPYING...
COPY OF FILE "qmargins.h.40A903CBD168D808.idx" IS FINISHED.
Current time after last join: 1685816556 seconds, 512962 microseconds
Total time passed to copy files: 0 seconds, 913215 microseconds
Total of 1961190 bytes were copied.
Total of 200 files were copied.
END OF MAIN
==3097==
==3097== HEAP SUMMARY:
==3097==     in use at exit: 0 bytes in 0 blocks
==3097==   total heap usage: 19 allocs, 19 frees, 173,888 bytes allocated
==3097==
==3097== All heap blocks were freed -- no leaks are possible
==3097==
==3097== For lists of detected and suppressed errors, rerun with: -s
==3097== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cabanel@MSI:/mnt/c/Users/Mert/Desktop$
```

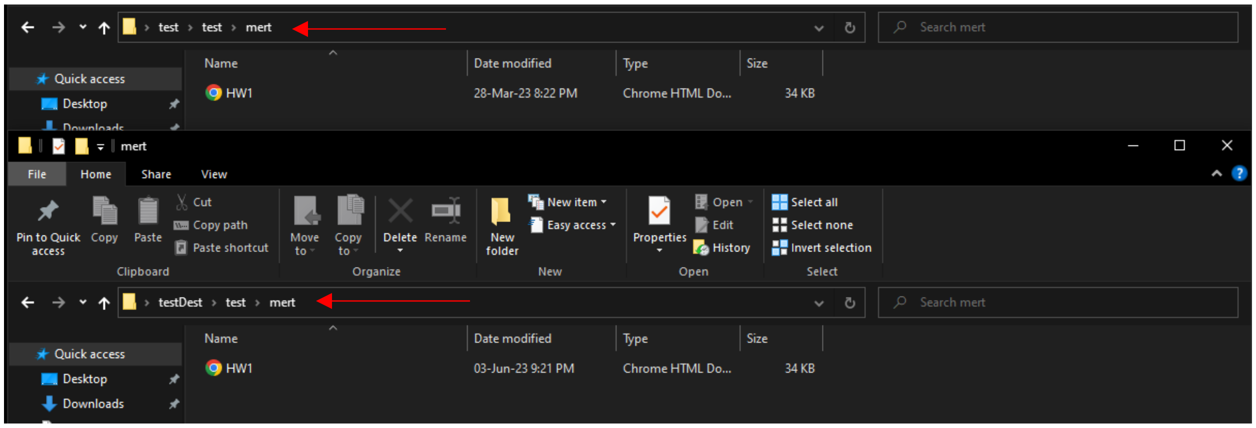
Only 200 files are copied. If we do same thing without interruption:



```
COPY OF FILE "test2.txt" IS FINISHED.
COPY OF FILE "CSE 351-S23-20230526_140123-Meeting Recording.mp4" IS FINISHED.
Current time after last join: 1685816633 seconds, 973308 microseconds
Total time passed to copy files: 17 seconds, 697350 microseconds
Total of 328767901 bytes were copied.
Total of 1481 files were copied.
END OF MAIN
==3111==
==3111== HEAP SUMMARY:
==3111==     in use at exit: 0 bytes in 0 blocks
==3111==   total heap usage: 37 allocs, 37 frees, 764,576 bytes allocated
==3111==
==3111== All heap blocks were freed -- no leaks are possible
==3111==
==3111== For lists of detected and suppressed errors, rerun with: -s
==3111== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
cabanel@MSI:/mnt/c/Users/Mert/Desktop$
```

1481 files are copied.

Subdirectories should be copied correctly:



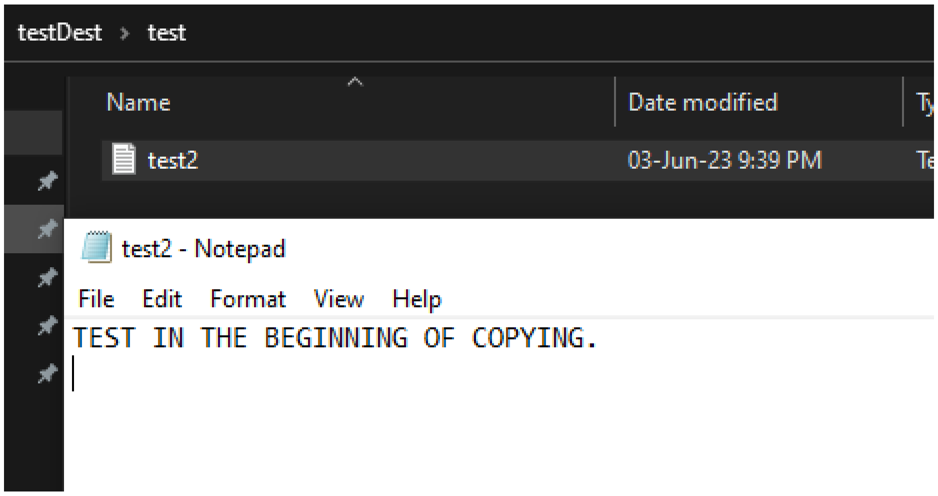
Name	Date modified	Type	Size
HW1	28-Mar-23 8:22 PM	Chrome HTML Do...	34 KB

Name	Date modified	Type	Size
HW1	03-Jun-23 9:21 PM	Chrome HTML Do...	34 KB

Same directory hierarchy and we have same 34 KB HW1.pdf file.

<u>If there is same file name, it should be overwritten:</u>

Before copying:



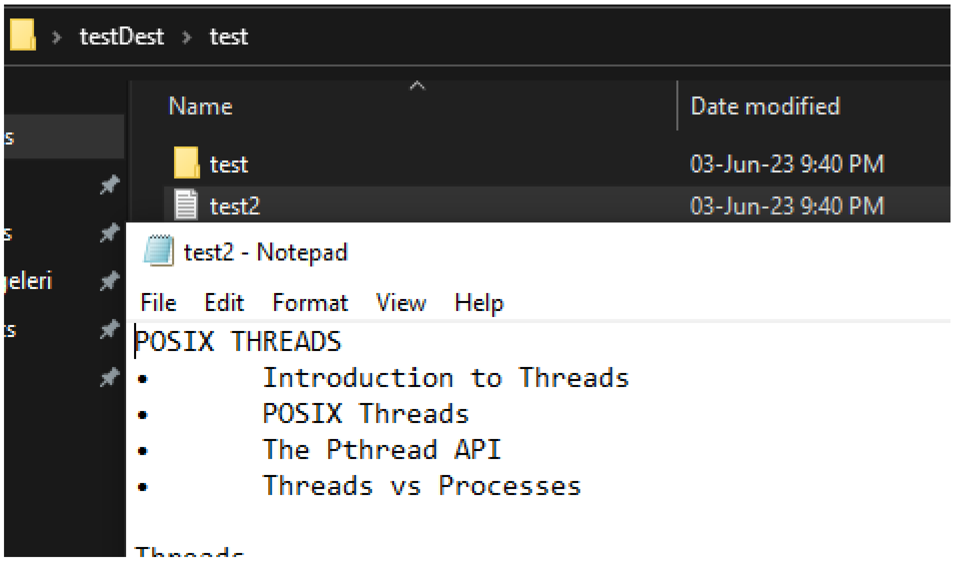
testDest › test

Name	Date modified
test2	03-Jun-23 9:39 PM

test2 - Notepad

```
TEST IN THE BEGINNING OF COPYING.
```

After copying:



testDest › test

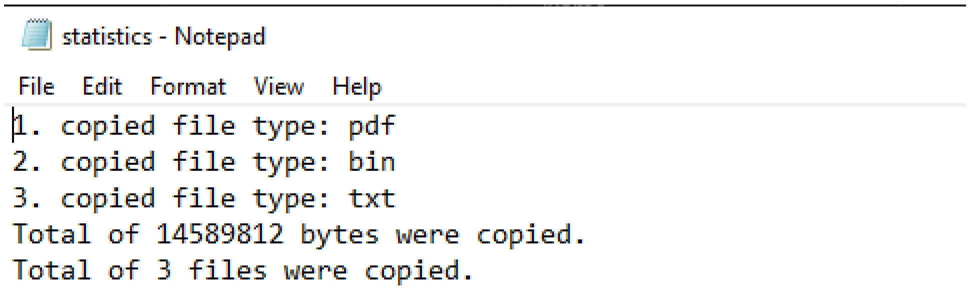
Name	Date modified
test	03-Jun-23 9:40 PM
test2	03-Jun-23 9:40 PM

test2 - Notepad

```
POSIX THREADS
•       Introduction to Threads
•       POSIX Threads
•       The Pthread API
•       Threads vs Processes
```

Statistics should be kept:

All the statistics are kept in the file named “statistics.txt”:



```
1. copied file type: pdf
2. copied file type: bin
3. copied file type: txt
Total of 14589812 bytes were copied.
Total of 3 files were copied.
```