TUBA TOPRAK

161044116 -Hw5

SYSTEM PROGRAMMING

It uses a producer-consumer model to copy files from a source directory to a destination directory, with multiple consumer threads performing the copying operation concurrently.

The necessary header files and libraries are included.

Global variables are declared, including buffer_size (to control the number of threads in the thread pool), num_consumer_threads (number of consumer threads), thread_pool (an array to store thread IDs), thread_availability (an array to track the availability of threads), and main_thread (ID of the main thread).

Mutex variables buffer_mutex and output_mutex are defined to synchronize access to shared resources.

The Files struct is defined to hold information about the source and destination directories, as well as the availability of a thread.

The err_sys function is a helper function to handle system errors.

The get_elapsed_time function calculates the elapsed time between two struct timeval timestamps

The ignore_sigint function is a signal handler for SIGINT (interrupt signal, usually triggered by pressing Ctrl+C). It cancels all consumer threads, frees memory, and exits gracefully.

The delete_directory function is a recursive function to delete a directory and its contents.

The customer function is responsible for copying a file from the source directory to the destination directory.

The customer_thread function is the entry point for consumer threads. It calls the customer function to copy a file and updates the thread availability status.

The producer function is the main logic for traversing the source directory and spawning consumer threads to copy files.

The producer_thread function is the entry point for the producer thread. It invokes the producer function and waits for all consumer threads to complete.

In the main function, command-line arguments are parsed to set the buffer size, number of consumer threads, source directory, and destination directory.

The ignore_sigint function is registered as the signal handler for SIGINT.

Validation is performed on the command-line arguments to ensure positive integer values for the buffer size and number of consumer threads.

Memory is allocated for the thread availability and thread pool arrays.

The thread availability array is initialized to 0 (indicating availability).

The destination directory is deleted (if it exists) and created anew.

Timing is started using gettimeofday().

The buffer mutex and output mutex are initialized.

The main thread creates the producer thread, which starts the file copying process.

The main thread waits for the producer thread to complete.

The mutexes are destroyed. Timing is ended, and the elapsed time is calculated and displayed. Memory is freed.

multithreading is used to improve the performance of the file copying process by leveraging concurrent execution. The program utilizes a producer-consumer model, where the producer thread traverses the source directory and spawns multiple consumer threads to perform the file copying operation.

Here's how the multithreading is implemented:

Thread Pool: The program creates a pool of consumer threads, specified by the num_consumer_threads variable. The thread pool is represented by the thread_pool array, which stores the thread IDs.

Thread Availability: To manage the availability of consumer threads, the thread_availability array is used. Each element of the array corresponds to a consumer thread, and a value of 0 indicates that the thread is available for processing a file. Initially, all threads are marked as unavailable (0).

Buffer Mutex: The buffer_mutex is a mutex (mutual exclusion) variable used to synchronize access to the thread_availability array. It ensures that only one thread at a time can check and update the availability status.

Output Mutex: The output_mutex is another mutex variable used to synchronize access to the output, such as printing the copied file information. It ensures that multiple threads don't interfere with each other when writing to the output.

Producer Thread: The main thread acts as the producer thread. It invokes the producer function, which recursively traverses the source directory. When it encounters a file, it checks for an available consumer thread using the thread_availability array. If an available thread is found, it marks it as unavailable, creates a structure (Files) to hold the file information, and passes it to the consumer thread using the pthread_create function. The producer thread then waits for all consumer threads to complete using pthread_join.
Consumer Threads: The consumer threads are responsible for copying files from the source directory to the destination directory. Each consumer thread executes the customer_thread function. Inside this function, the customer function is called to perform the file copy operation. Access to the shared resources (thread availability and output) is protected using mutex locks (pthread_mutex_lock and pthread_mutex_unlock).

Controls:

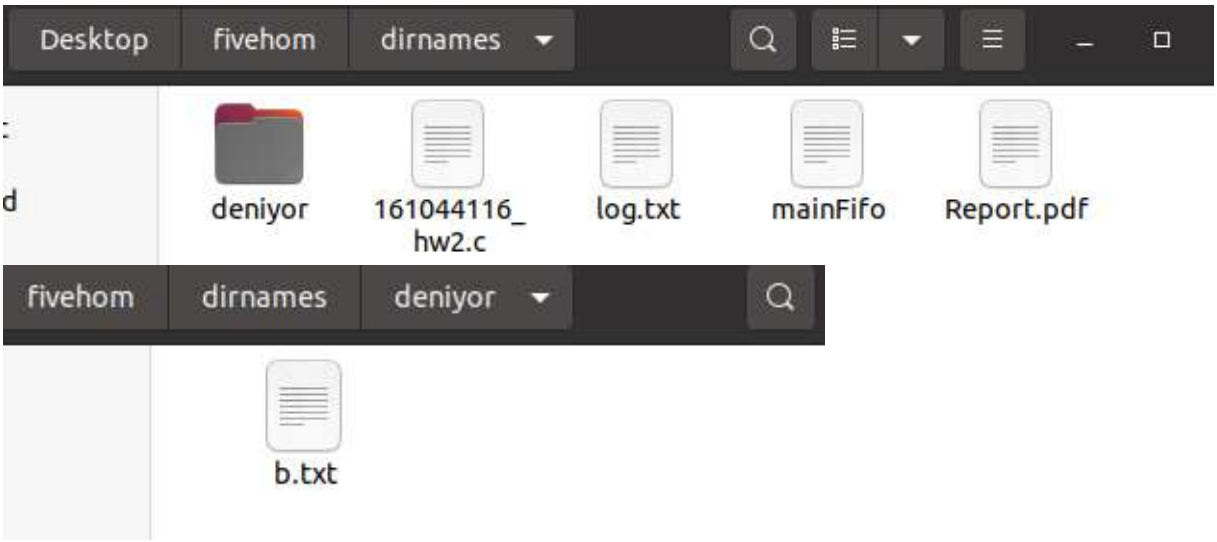
```
toprak@toprak-virtual-machine:~/Desktop/fivehom$ ps -aux | grep "Z"
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
toprak     23165  0.0  0.0  11776   652 pts/0    S+   22:47   0:00 grep --color=auto Z
```

Output:

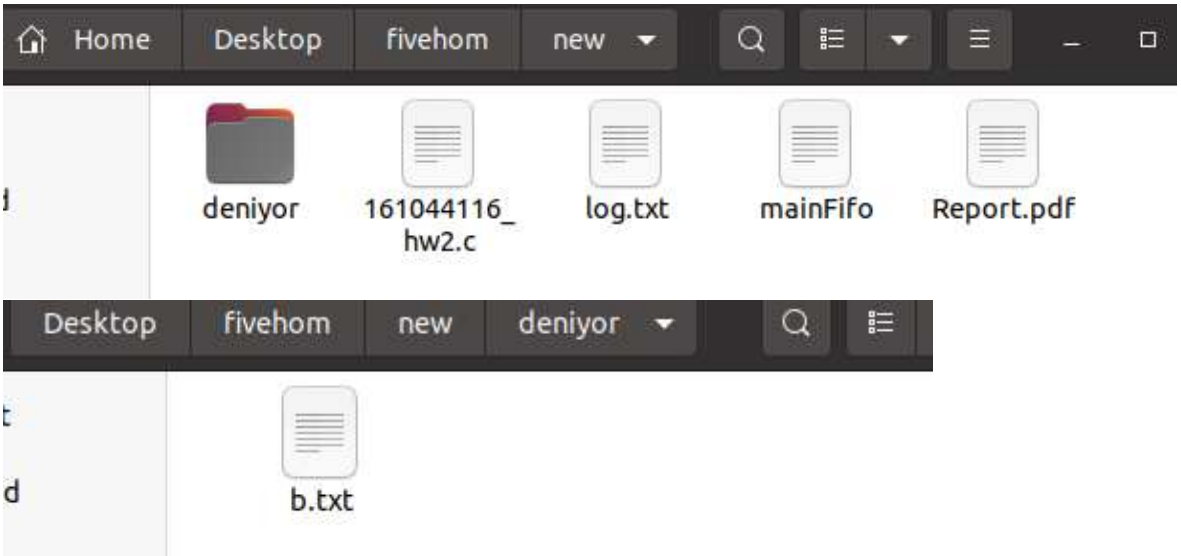
```
toprak@toprak-virtual-machine:~/Desktop/fivehom$ ./pCp 100 10 dirnames new

Copying from dirnames folder to new folder.
Copied Folder: new/deniyor
Copied File: new/deniyor/b.txt
Copied File: new/log.txt
Copied Fifo: new/mainFifo
Copied File: new/Report.pdf
Copied File: new/161044116_hw2.c
Elapsed time: 0.002077 seconds
Finished..
toprak@toprak-virtual-machine:~/Desktop/fivehom$
```

Source:



Destination:



Observations:

As the buffer size increased, the elapsed time decreased. This indicates that a larger buffer allows for more parallelism and reduces the time spent waiting for available consumer threads.

Similarly, increasing the number of consumer threads resulted in decreased elapsed time. More threads allow for higher concurrency, leading to faster file copying.

Conclusion:

The multithreaded file copying utility demonstrated improved performance compared to a single-threaded approach. The experiments revealed that larger buffer sizes and a higher number of consumer threads yielded better results in terms of elapsed time.