CSE 333 - OPERATING SYSTEMS
PROJECT #3
“MULTI-THREADED PROGRAM”

Melisa DÖNMEZ-150116030
Rıdvan GÜLCÜ-150117508

This program gets four arguments like as:
 $ ./<executable_name> “-d” <txt file name> “-n” <read thread number> <upper thread number> <replace thread number> <write thread number>

Reader threads read one first non-readed line and keep this line on array. Reader threads can run simultaneously so we keep indexes on global for this read and store operations.We must the protect this shared variables.readIndexMutex is protect the read index and storeIndexMutex protect the store index for array.

Upper threads and replace threads are wait until added the lines to array. replaceSemaphore for control the number of elements that have not been replaced in the array and upperSemaphore for control the number of elements that have not been uppered in the array.Both threads ara run simultaneously but must run on different indexes.So we control the index for this operations and we not allow for simultaneously replace and upper operation same index at same time. replaceIndexMutex and upperIndexMutex protect global index variables and mutexes in the indexChangeMutex array is not allowed one more than process same index at the same time.

Writer threads are wait until upper and replace process done for at least one index. replaceWriteSemaphore for control the number of the elements which replaced already. upperWriteSemaphore for control the number of the elements which uppered already

There could be only one writer Thread the can write to the file.So we protect all parts of write process. writeReadMutex is provides it.

Readers can not enter if any running writer process and vice versa. But reader can enter if any running reader process.So we must the keep counter of currently run reader process on global and we must protect this variable. readerCountMutex is protect this variable.And we apply classical reader writer problem solution for this problem. When Reader is request check the counter of readers if the are no reader inside wait the writeReadMutex .When Reader exit check again counter of reader. if the are no reader inside signal to writeReadMutex.

First of all we checked the arguments with checkArguments function which we call beginning of the main function. This function checks whether the user’s input is correct. First check the count of

argument.It must be equal 8. After find the index of -n -d options. If there is no any of these terminate program.After that, check the index of this options.If index of -d and -n location is not appropriate display an error message and terminate the program.After all check the file is exist and given thread numbers.
If inputs are correct then assigns the txt name and thread counts to global variables.

We created structure for each thread to keep information about threads. In these structures, we have kept the index of threads and lines as a lineData type. Also lineData is a structure which we created to keep information about lines. In the main function we created array of these structure fo each type of threads. We used these arrays in the create threads function for every threads. In general, we have assigned an indew for the thread and created the number of threads as many as the given number of inputs in the thread creation functions. Finally we called the functions that the threads do their tasks inside the pthread_create.

In the readFile function read thread start reading the txt file and keep the data in the lines array which we created this array in the global. But threads should not read the same line. Because of that this part critical section. To prevent thread from reading the same line we used the mutex.We writed readIndexMutex mutex in the global. Where we increased the line number to be read, we locked this mutex and then unlocked it. After specifying the line number to read called readSpecificLine function. The task of this function is to find and return that line in the txt according to the line number to be read. Thread prints the information of the job on the screen after reading the line returned from this function. The read line should be stored in the lines array after the reading event ends. But threads should not store in the same index in the array. To prevent this we created storeIndexMutex mutex in the global. Where we increased the index number which stored in array, we locked this mutex and then unlocked it. Read threads after finishing the task joinReadThreads function is called to wait for read threads to terminate. At the end of read process add one to replaceSemaphore and upperSemaphore.

runUpper wait the upperSemaphore and get index from global with mutex. After decide index covert the string on array depend on this index.Conversion part is just protect with replaceIndexMutex.So threads run simultaneously for different indexes.At the end of conversion add one the upperWriteSemaphore with sem_post.

runReplace works with exactly same idea with runUpper method.Wait the replaceSemaphore and at the end of upper process add one the replaceWriteSemaphore with sem_post.

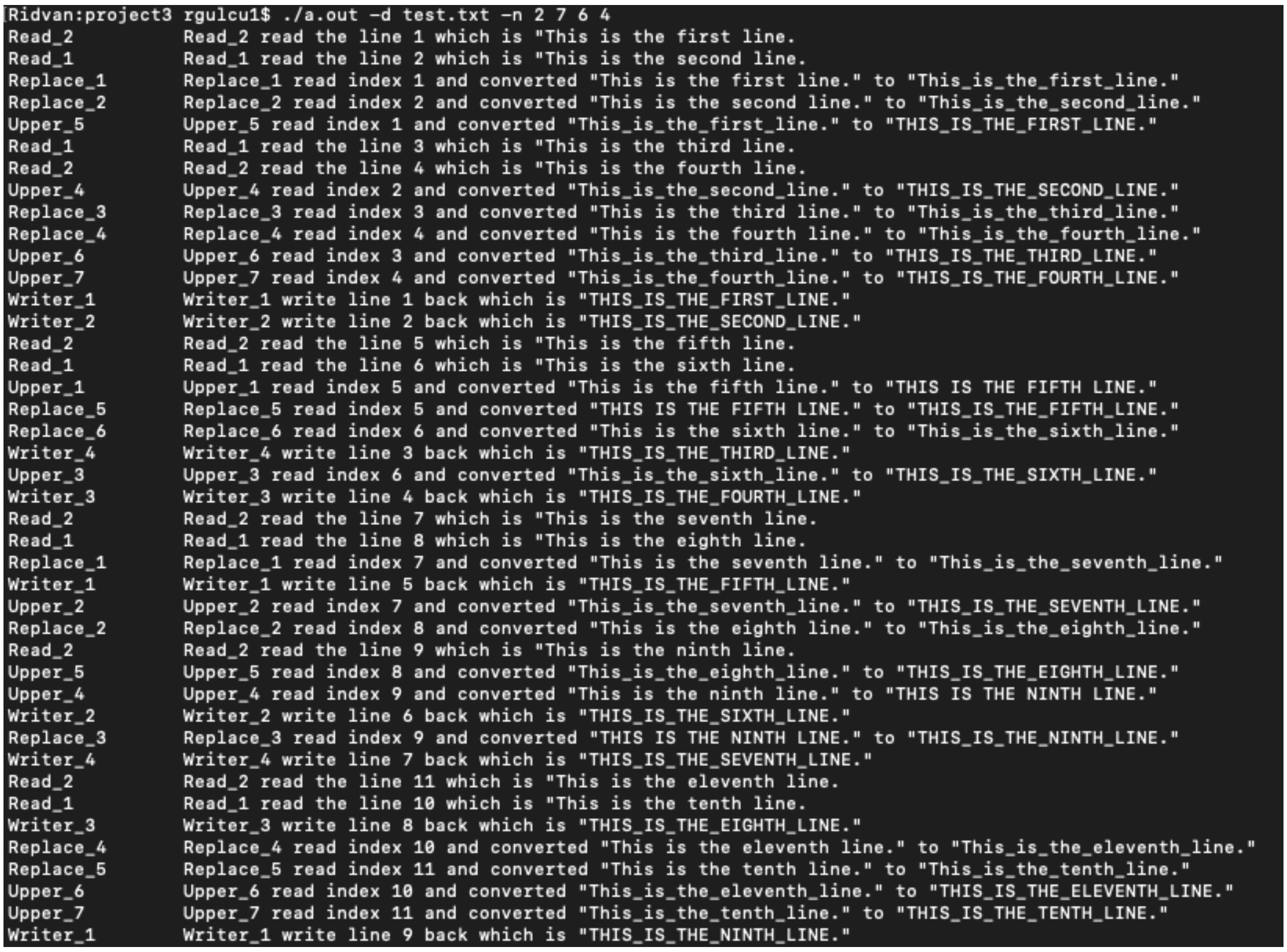
runWrite wait the both replaceWriteSemaphore and upperWriteSemaphore.After that takes index from global and update the file depend on this index.

After finishing upper and replace threads jobs txt file data should be updated with using the final version of the lines array. For the write part we created createWriteThreads function in the main. This function creates the write threads according to write thread number which is taken input from user. Inside this function, we have assigned write thread indexes that we hold in the writeThreadData structure. Also we used pthread_create function to create a thread inside this function and called the *runWrite function for all write threads. Each write thread should read the data in the different index from the lines array and write it to the correct line in the txt file. To prevent threads from reading the same index from array we created writeIndexMutex mutex in the global. In the section where the index of the array to be read is determined and increased, we locked the mutex and then we uncloked it.

After deciding the index of the array to be read in the runWrite function, we called the writeFile-SpecificIndex function to write to txt. Inside the writeFileSpecificIndex prevent write threads from updating to the same line in the txt file. For this we used writeFileMutex which we created in the global. This mutex must be locked at the beginning of the writeFileSpecificIndex function and unlocked at the end. In general, this function takes the desired lines as parameter and makes the change with using fseek and fprinf functions.This period is repeated until the write threads change all lines. Write threads after finishing the task joinWriteThreads function is called to wait for read threads to terminate.

Example Outputs:

A part of output for 50 line test.txt file



```
[Ridvan:project3 rgulcu1$ ./a.out -d test.txt -n 2 7 6 4
Read_2          Read_2 read the line 1 which is "This is the first line.
Read_1          Read_1 read the line 2 which is "This is the second line.
Replace_1       Replace_1 read index 1 and converted "This is the first line." to "This_is_the_first_line."
Replace_2       Replace_2 read index 2 and converted "This is the second line." to "This_is_the_second_line."
Upper_5         Upper_5 read index 1 and converted "This_is_the_first_line." to "THIS_IS_THE_FIRST_LINE."
Read_1          Read_1 read the line 3 which is "This is the third line.
Read_2          Read_2 read the line 4 which is "This is the fourth line.
Upper_4         Upper_4 read index 2 and converted "This_is_the_second_line." to "THIS_IS_THE_SECOND_LINE."
Replace_3       Replace_3 read index 3 and converted "This is the third line." to "This_is_the_third_line."
Replace_4       Replace_4 read index 4 and converted "This is the fourth line." to "This_is_the_fourth_line."
Upper_6         Upper_6 read index 3 and converted "This_is_the_third_line." to "THIS_IS_THE_THIRD_LINE."
Upper_7         Upper_7 read index 4 and converted "This_is_the_fourth_line." to "THIS_IS_THE_FOURTH_LINE."
Writer_1        Writer_1 write line 1 back which is "THIS_IS_THE_FIRST_LINE."
Writer_2        Writer_2 write line 2 back which is "THIS_IS_THE_SECOND_LINE."
Read_2          Read_2 read the line 5 which is "This is the fifth line.
Read_1          Read_1 read the line 6 which is "This is the sixth line.
Upper_1         Upper_1 read index 5 and converted "This is the fifth line." to "THIS IS THE FIFTH LINE."
Replace_5       Replace_5 read index 5 and converted "THIS IS THE FIFTH LINE." to "THIS_IS_THE_FIFTH_LINE."
Replace_6       Replace_6 read index 6 and converted "This is the sixth line." to "This_is_the_sixth_line."
Writer_4        Writer_4 write line 3 back which is "THIS_IS_THE_THIRD_LINE."
Upper_3         Upper_3 read index 6 and converted "This_is_the_sixth_line." to "THIS_IS_THE_SIXTH_LINE."
Writer_3        Writer_3 write line 4 back which is "THIS_IS_THE_FOURTH_LINE."
Read_2          Read_2 read the line 7 which is "This is the seventh line.
Read_1          Read_1 read the line 8 which is "This is the eighth line.
Replace_1       Replace_1 read index 7 and converted "This is the seventh line." to "This_is_the_seventh_line."
Writer_1        Writer_1 write line 5 back which is "THIS_IS_THE_FIFTH_LINE."
Upper_2         Upper_2 read index 7 and converted "This_is_the_seventh_line." to "THIS_IS_THE_SEVENTH_LINE."
Replace_2       Replace_2 read index 8 and converted "This is the eighth line." to "This_is_the_eighth_line."
Read_2          Read_2 read the line 9 which is "This is the ninth line.
Upper_5         Upper_5 read index 8 and converted "This_is_the_eighth_line." to "THIS_IS_THE_EIGHTH_LINE."
Upper_4         Upper_4 read index 9 and converted "This is the ninth line." to "THIS IS THE NINTH LINE."
Writer_2        Writer_2 write line 6 back which is "THIS_IS_THE_SIXTH_LINE."
Replace_3       Replace_3 read index 9 and converted "THIS IS THE NINTH LINE." to "THIS_IS_THE_NINTH_LINE."
Writer_4        Writer_4 write line 7 back which is "THIS_IS_THE_SEVENTH_LINE."
Read_2          Read_2 read the line 11 which is "This is the eleventh line.
Read_1          Read_1 read the line 10 which is "This is the tenth line.
Writer_3        Writer_3 write line 8 back which is "THIS_IS_THE_EIGHTH_LINE."
Replace_4       Replace_4 read index 10 and converted "This is the eleventh line." to "This_is_the_eleventh_line."
Replace_5       Replace_5 read index 11 and converted "This is the tenth line." to "This_is_the_tenth_line."
Upper_6         Upper_6 read index 10 and converted "This_is_the_eleventh_line." to "THIS_IS_THE_ELEVENTH_LINE."
Upper_7         Upper_7 read index 11 and converted "This_is_the_tenth_line." to "THIS_IS_THE_TENTH_LINE."
Writer_1        Writer_1 write line 9 back which is "THIS_IS_THE_NINTH_LINE."
```