To the Reader

In a normal execution (with a default format), our code works completely. These explanations are for the things that is general or something we do not mentioned while commenting.

When executing the code, you should follow the default format(ex: ./program -d file.txt -n 1 2 3 4)

General Structure

- We stored almost all necessary information in global variables. In this way, we do not deal with referencing much. Also, we used static arrays for again, simplicity.

- We first define line and array sizes. Then, create 'lines' array with these parameters.

```c
# define MAX_LINE_LENGTH 128
# define ARRAY_SIZE 64

char lines[ARRAY_SIZE][MAX_LINE_LENGTH]; // array of string
```

- We store file name in a char pointer named 'fileName' and size of the file (number of lines) in an integer variable named 'fileSize'. We have a function for finding the number of lines in the file and update the fileSize parameter. We use checkFileSize function just once, at the beginning of main function.

```c
char* fileName;
char fileSize;
```

- We have global variables for number of threads of each kind. Also, we have indexes for each operation. For example, we have numReadThreads storing the number of read threads.
  Also, we have readIndex. If readIndex is 1, it means we have read one line from the file and stored it in the global 'lines' array. If it is equal to the fileSize, it means we read all the lines and stored them in 'lines'.
- For a thread to cancel out, related index must reach the file size. For example, If fileSize = 10, when readIndex reach 10, all read threads cancel.

Main Function

- In main function, we first parse the arguments. In here, we are checking errors via argc, not for -d or -n. It means we must give the parameters in the right format.

```c
if(argc != 8){
printf("You should enter the argument as follows:\n"
    "./ProgramName -d TxtFileName -n numberOfReadThreads"
    " numberOfUpperThreads numberOfReplaceThreads numberOfWriteThreads");
    return 1;
}
```

- After parsing, we create all the threads based on the inputs we gave and joined them for main to wait for all threads to execute.

Read Function of Read Thread

- In readFunc, we first lock the system for synchronization. Then, we check if we can do a read operation by checking readIndex. If it is equal to fileSize, it means we are done with reading. In this case, we unlock the system and cancel the thread. All read threads come to this situation eventually.
- If we are allowed to do a read operation, we open the file, read and throw away the lines until we find the right line. For this purpose, we consider readIndex. When we reach the right line, we add this to our global 'lines' array. Then, we increment the readIndex by 1.
- After these, we close the file and unlock the system.

Upper & Replace Threads

- These two types of theads have the same logic. We first lock the system and check for if we are allowed to do an operation. This checking was pain in the neck. The logic is as follows:

```
int shouldStop=1;                      // this is the variable for stopping the whole operation

if(readIndex==fileSize){               // if we have read all the files or upperIndex
        shouldStop=0;                  // is still smaller than readIndex, we continue
}
else if(upperIndex < readIndex){
        shouldStop=0;
}


if(shouldStop){
        pthread_mutex_unlock(&systemLock);// if we are not allowed to do 'upper' operation,
        continue;                         // we unlock the system and wait for an appropriate situation
}

if(upperIndex == fileSize){                  // if we have made all the lines in upper case letters,
        pthread_mutex_unlock(&systemLock);   // we break and exit.
        break;
}
```

- If we are okay to go, we first get the line from the lines array by considering upperIndex (or replaceIndex). Then, we do the operation in a local char pointer. After doing this, we print the result, increment the index by one and unlock the system.
- All the things except 'upper operation' and 'replace operation' are the same for upper and replace threads.

Write Function of Write Thread

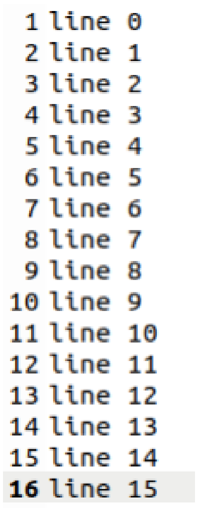
- At start, we lock the system and check if we are allowed to do a write operation. The logic is as follows:

```
pthread_mutex_lock(&systemLock); // for write operation, we should be sure that related index is done
                                 // with read, replace and write operations. After that, we can continue.
int shouldStop=1;
if(readIndex==fileSize && upperIndex==fileSize && replaceIndex==fileSize){
        shouldStop = 0;
}
else if(writeIndex < readIndex && writeIndex < replaceIndex && writeIndex < upperIndex){
        shouldStop = 0;
}

if(shouldStop){
        pthread_mutex_unlock(&systemLock);
        continue;
}

if(writeIndex == fileSize){
        pthread_mutex_unlock(&systemLock);
        break;
}
```

- If we are allowed to do a write operation, we first open the file in r+ mode (to both read and write) and start to read. When we reach the index that we want to write on, we stop reading and write the related index to the file. Then, we close the file and unlock the system.

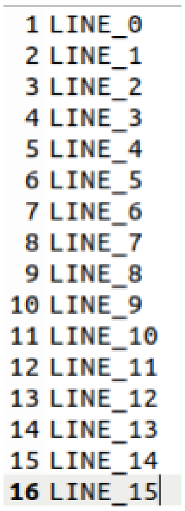
Execution Example

First

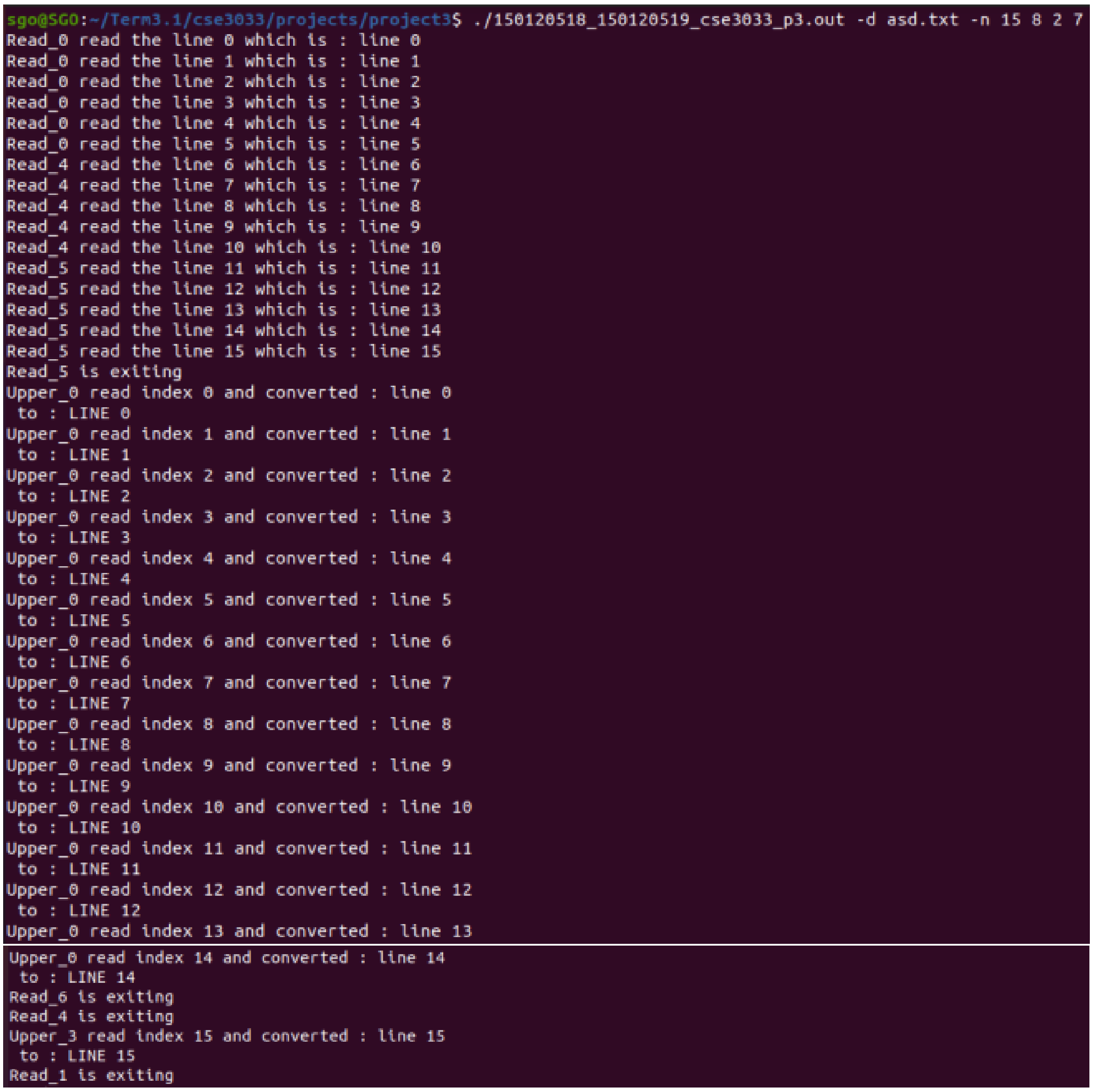


```
line 0
line 1
line 2
line 3
line 4
line 5
line 6
line 7
line 8
line 9
line 10
line 11
line 12
line 13
line 14
line 15
```

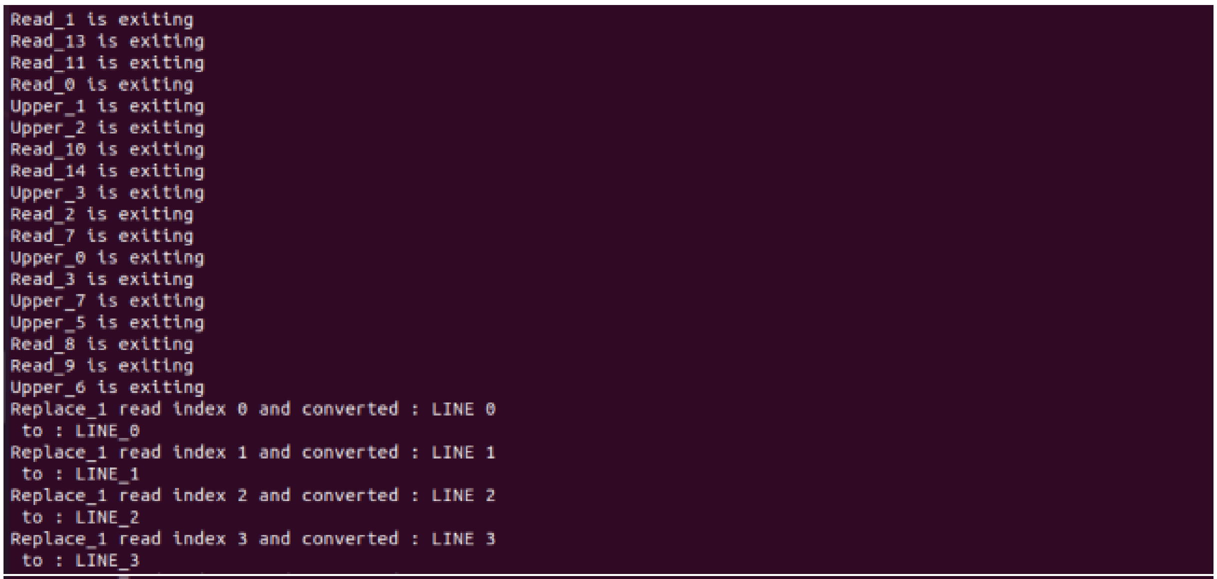
Second



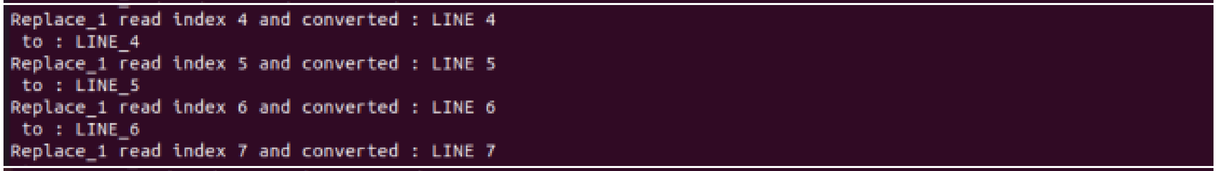
```
LINE_0
LINE_1
LINE_2
LINE_3
LINE_4
LINE_5
LINE_6
LINE_7
LINE_8
LINE_9
LINE_10
LINE_11
LINE_12
LINE_13
LINE_14
LINE_15
```



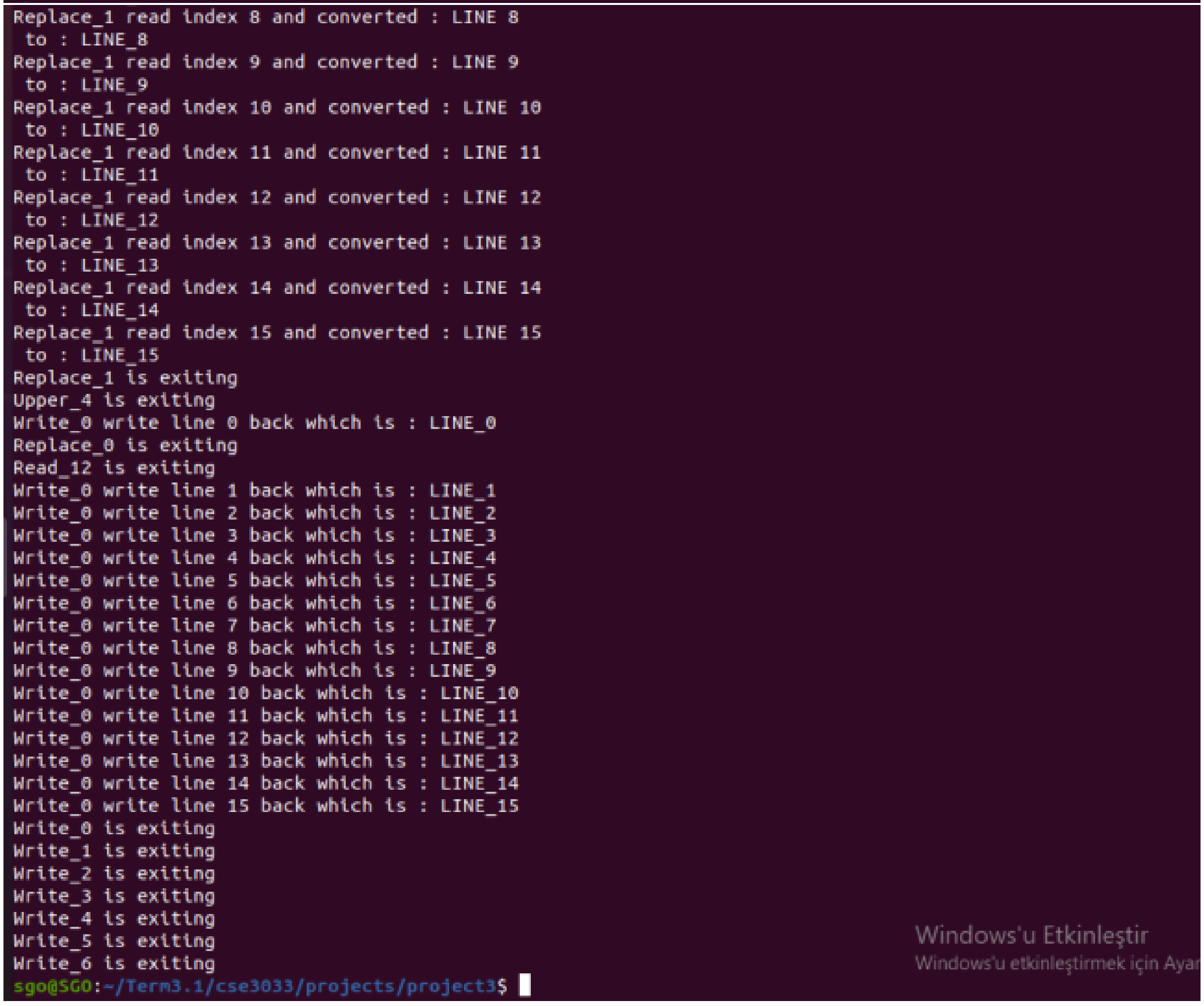
```
sgo@SGO:~/Term3.1/cse3033/projects/project3$ ./150120518_150120519_cse3033_p3.out -d asd.txt -n 15 8 2 7
Read_0 read the line 0 which is : line 0
Read_0 read the line 1 which is : line 1
Read_0 read the line 2 which is : line 2
Read_0 read the line 3 which is : line 3
Read_0 read the line 4 which is : line 4
Read_0 read the line 5 which is : line 5
Read_4 read the line 6 which is : line 6
Read_4 read the line 7 which is : line 7
Read_4 read the line 8 which is : line 8
Read_4 read the line 9 which is : line 9
Read_4 read the line 10 which is : line 10
Read_5 read the line 11 which is : line 11
Read_5 read the line 12 which is : line 12
Read_5 read the line 13 which is : line 13
Read_5 read the line 14 which is : line 14
Read_5 read the line 15 which is : line 15
Read_5 is exiting
Upper_0 read index 0 and converted : line 0
 to : LINE 0
Upper_0 read index 1 and converted : line 1
 to : LINE 1
Upper_0 read index 2 and converted : line 2
 to : LINE 2
Upper_0 read index 3 and converted : line 3
 to : LINE 3
Upper_0 read index 4 and converted : line 4
 to : LINE 4
Upper_0 read index 5 and converted : line 5
 to : LINE 5
Upper_0 read index 6 and converted : line 6
 to : LINE 6
Upper_0 read index 7 and converted : line 7
 to : LINE 7
Upper_0 read index 8 and converted : line 8
 to : LINE 8
Upper_0 read index 9 and converted : line 9
 to : LINE 9
Upper_0 read index 10 and converted : line 10
 to : LINE 10
Upper_0 read index 11 and converted : line 11
 to : LINE 11
Upper_0 read index 12 and converted : line 12
 to : LINE 12
Upper_0 read index 13 and converted : line 13
Upper_0 read index 14 and converted : line 14
 to : LINE 14
Read_6 is exiting
Read_4 is exiting
Upper_3 read index 15 and converted : line 15
 to : LINE 15
Read_1 is exiting
```



```
Read_1 is exiting
Read_13 is exiting
Read_11 is exiting
Read_0 is exiting
Upper_1 is exiting
Upper_2 is exiting
Read_10 is exiting
Read_14 is exiting
Upper_3 is exiting
Read_2 is exiting
Read_7 is exiting
Upper_0 is exiting
Read_3 is exiting
Upper_7 is exiting
Upper_5 is exiting
Read_8 is exiting
Read_9 is exiting
Upper_6 is exiting
Replace_1 read index 0 and converted : LINE 0
 to : LINE_0
Replace_1 read index 1 and converted : LINE 1
 to : LINE_1
Replace_1 read index 2 and converted : LINE 2
 to : LINE_2
Replace_1 read index 3 and converted : LINE 3
 to : LINE_3
```



```
Replace_1 read index 4 and converted : LINE 4
 to : LINE_4
Replace_1 read index 5 and converted : LINE 5
 to : LINE_5
Replace_1 read index 6 and converted : LINE 6
 to : LINE_6
Replace_1 read index 7 and converted : LINE 7
```



```
Replace_1 read index 8 and converted : LINE 8
 to : LINE_8
Replace_1 read index 9 and converted : LINE 9
 to : LINE_9
Replace_1 read index 10 and converted : LINE 10
 to : LINE_10
Replace_1 read index 11 and converted : LINE 11
 to : LINE_11
Replace_1 read index 12 and converted : LINE 12
 to : LINE_12
Replace_1 read index 13 and converted : LINE 13
 to : LINE_13
Replace_1 read index 14 and converted : LINE 14
 to : LINE_14
Replace_1 read index 15 and converted : LINE 15
 to : LINE_15
Replace_1 is exiting
Upper_4 is exiting
Write_0 write line 0 back which is : LINE_0
Replace_0 is exiting
Read_12 is exiting
Write_0 write line 1 back which is : LINE_1
Write_0 write line 2 back which is : LINE_2
Write_0 write line 3 back which is : LINE_3
Write_0 write line 4 back which is : LINE_4
Write_0 write line 5 back which is : LINE_5
Write_0 write line 6 back which is : LINE_6
Write_0 write line 7 back which is : LINE_7
Write_0 write line 8 back which is : LINE_8
Write_0 write line 9 back which is : LINE_9
Write_0 write line 10 back which is : LINE_10
Write_0 write line 11 back which is : LINE_11
Write_0 write line 12 back which is : LINE_12
Write_0 write line 13 back which is : LINE_13
Write_0 write line 14 back which is : LINE_14
Write_0 write line 15 back which is : LINE_15
Write_0 is exiting
Write_1 is exiting
Write_2 is exiting
Write_3 is exiting
Write_4 is exiting
Write_5 is exiting
Write_6 is exiting
sgo@SGO:~/Term3.1/cse3033/projects/project3$
```