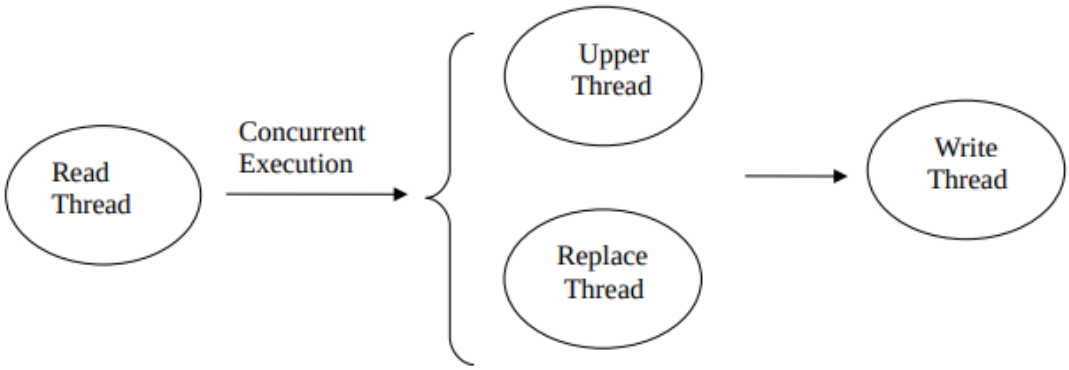
In this assignment, you will write a program that uses threads and synchronization to read from a file, modify its contents and write to the same file. You will implement a program with four types of threads, structured like this:



- The Read Thread will read from a file line by line and store every line in an array.
- The Upper Thread will read from the array, convert the string to uppercase letters and write back to the same index of the array.
- The Replace Thread will read from the array, replace every space with the underscore character and write back to the same index of the array.
- The Write Thread will read from the array and store it in the text file, each line in the current place.

Program Details

- The user will execute the program specifying the number of threads to be created of each kind. You will then create the specified number of threads of each type.
- The Read Threads will read from the file. However, they first should be assigned to which line of the file they should read. Each Read Thread must acquire a unique line number. Note that you must make sure no two Read Threads read the same line from the file. After they read their assigned line, they must store the line they read in an array (or any other data structure you chose). If a Read Thread reads the 0th line of the text file, it must store it in the 0th index of the array.
- After there is at least one entry in the array, the Upper Threads will read the array indices one by one and change the content to uppercase. Note that all Upper Threads must read a different array index. Note also that Upper Threads should work concurrently with Replace Threads.
- After there is at least one entry in the array, the Replace Threads will read the array indices one by one and substitute all the spaces with underscore characters. Note that all Replace Threads must read a different array index. Note also that Replace Threads should work concurrently with Upper Threads, which means that a line can be modified first by a Replace Thread and then an Upper Thread or first by an Upper Thread and then a Replace Thread. However, no multiple modifications can be done at the same time. You must make sure if one of the Upper or Replace Threads is working on an index, the other one should not be able to.

- After both modifications are done with at least one array index, the Writer Threads will update the file's appropriate line with the array's corresponding entry. Note that there could be only one Writer Thread that can write to the file. Note also that you must make sure no same writer thread updates the same line of the file.
- The main thread is responsible for creating all the threads and waiting for them. Note that, all the threads must exist in the system at the same time, and you must work on synchronizing them.
- All the threads must print information about their job on the screen (see the example execution below).
- Note that there is a global queue requested and the queue can be accessed by many threads of different types, so synchronization is necessary for the queue. This can be achieved by using semaphores.
- The global queue must be created the first time a thread needs to use by that thread.
- Preserving consistency and preventing deadlocks are major issues to be considered.
- When a thread is done with its job, it must check if there are more jobs to do. (e.g.: When a Replace Thread is done with one line, it must check the array once more to see if there are more indices that are not modified by a Replace Thread yet.)
- Multiple simultaneous operations on different parts of the queue should be allowed in your solution.
- Your program will be executed as follows:
    - Example:

      ./project3.out -d deneme.txt -n 15 5 8 6
    - The -d option represents the name of the file you will read from (a simple text file), -n option represents the number of threads to be created of each type (Read, Upper, Replace, and Write, respectively).
- Your program should produce output.
    - Example:

<Thread-type and ID>	<Output>
Read_1	Read_1 read the line 1 which is "This is an example line."
Read_2	Read_2 read the line 2 which is "This is another line."
Upper_2	Upper_2 read index 2 and converted "This is another line." to "THIS IS ANOTHER LINE."
Replace_2	Replace_2 read index 1 and converted "This is an example line." to "This_is_an_example_line."
Upper_1	Upper_1 read index 1 and converted "This_is_an_example_line." to "THIS_IS_AN_EXAMPLE_LINE."
Replace_1	Replace_1 read index 2 and converted "THIS IS ANOTHER LINE." to "THIS_IS_ANOTHER_LINE."
Writer_1	Writer_1 write line 1 back which is "THIS_IS_AN_EXAMPLE_LINE."
Writer_3	Writer_3 write line 2 back which is "THIS_IS_ANOTHER_LINE."