CSE 3033 - OPERATING SYSTEMS
Report of Programming Assignment #3
150117011 – 150116049
Buse Batman – Göksel Tokur

This programming assignment was about writing a program that uses threads and synchronization to read from a file, modify its contents (turn to uppercase and replace every space character with underscore character) and write back to the same file.

For that we used four types of threads; read thread, uppercase thread, underscore thread and write thread.

Global:

-At first we defined two constants: LINESIZE and NUMBEROFLINES.

-We created a mutex and 4 semaphores (read, upper, replace and write semaphores).

-For the lines, we created a struct called lines_str to store a line's string value and 4 integers to check if that line has readed, uppercased, underscored or written.

-And we created some global variables: lines_str array of size LINESIZE, filename, fileindex, line counts, lineend and fileend for reading the file and storing the index of the processed line.

In main method:

-At first we checked the possible errors for running the code. If user doesn't supply valid number of arguments, valid characters or tries to give a filename which does not exist in that directory, then program prints error message about the correct usage.

-If there is no problem with the given command line arguments, we created the thread arrays with given thread numbers.

-We initalized the array of lines struct with null for the string and 0 for the integer values.

-We opened the file, user wants to make changes on. And we calculated the length of the file. Because those are global variables, we used these values in the other methods.

-We initialized the mutex and checked if there is an error with initialization.

-We initialized the 4 semaphores with given thread numbers.

-We created the threads with their specific methods, checked if any error occurred and then joined the threads.

-After everything finished, mutex has been destroyed.

In readFile method:

-Within a while loop, we opened the file with the given filename. And then checked if the file opened. If the file doesn't open, it prints error message. Or if it came to the end of the file, it returns; if not, continues.

-We called sem_wait with semaphore_queue_read and locked the mutex.

-And with fseek, we read the file with fileIndex and SEEK_SET. After every read operation, fileIndex increases with the length of the readed line.

-After reading the line, we copied the content of the line to the array and setted the value of readed to 1.

-After these read operations, we used sem_post and unlocked the mutex.

-Because these operations are in a while loop, a thread goes back and checks if there is any unreaded line until the conditions we explained at the top, are not satisfied.

In repToUpper method:

-Within a while loop, at first we checked if we came to the end of the file, if it is, it returns; if not, continues.

-If the current looking index of the array is not empty (readed) or not uppercased before, we started to do the operations.

-We called sem_wait with semaphore_queue_upper and locked the mutex.

-And we converted the string to uppercase with toupper function and set the value of uppered to 1.

-After these operations, we used sem_post and unlocked the mutex.

-Because these operations are in a while loop, a thread goes back and checks if there are any other lines not converted to uppercase.

In repToUnderscore method:

-Within a while loop, at first we checked if we came to the end of the file, if it is, it returns; if not, continues.

-If the current looking index of the array is not empty (readed) or not underscored before, we started to do the operations.

-We called sem_wait with semaphore_queue_replace and locked the mutex.

-And we checked every character whether it is space character. If it is we converted it to underscore character and set the value of underscored to 1.

-After these operations, we used sem_post and unlocked the mutex.

-Because these operations are in a while loop, a thread goes back and checks if there are any other lines not replaced.

In writeFile method:

-Within a while loop, at first we checked if we came to the end of the file, if it is, it returns; if not, continues.

-If the current looking index of the array is not empty (readed) or not written to the file before, we started to do the operations.

-We called sem_wait with semaphore_queue_write and locked the mutex.

-And we opened the same file with “r+” mode and wrote the processed lines back to the file.

-After these operations, we used sem_post and unlocked the mutex.

-Because these operations are in a while loop, a thread goes back and checks if there are any other lines not written to the file.