**CSE 3033 - OPERATING SYSTEMS**

**Programming Assignment # 3**

**Purpose:** In this assigment we were trying to implement multithreading in our third project in order to understand the workflow of opearating system efficiently.

**Problem:** The task that is requested from us is implementing four type of threads (read, upper,replace and write) in order to achieve synchronized programming. There may be multiple number of threads from each kind of threads. Read threads are supposed to read some data from txt file and parse it to an array or any kind of data structure. Upper threads are supposed to get data from that data structure and convert to uppercase. Samely replace threads are supposed to convert spaces to underscore and finally write threads are supposed to write the final result of data to same txt file that is given as input. So the main problem is synchronization among all of these threads. For example if there is no read data other threads could not do their job. We were trying to solve that problem most efficient and accurate way.

**Algorithm:** In our project we tried to implement some algorithm in order to achieve the problem we mentioned above.

**void \*init\_linked\_list(void \* number):**

Before talking about main function there is a init function which initialize a linked list with size of number of lines. In this function there is no parsing data from txt to linked list. It just initialize a linked list with null values.

**int main(int argc, char \*argv[]):**

In main function we created all necessary threads and initialized them. Also there is join function for all threads inside main function. We also initialize all semaphores inside main function in order to implement syncronization.

**typedef struct Threads{**

**pthread\_t thread;**

**int id;**

**}threads;**

Before continue with functions we would like to explain the Thread structure in order to understand the algorithm better. So we created an array of structure which is consist of a thread and its id. We gave the id according to their index inside threads[] array. Tnanks to that each thread will have a unique id. So the process of each id is according to index of line.

Lets say we have 10 lines of txt file and we have 2 read thread and for the first thread which its id is 0 it should be responsible from 10%2=id lines (0,2,4,6...10) and samely second array will responsible from 10%2=id lines (1,3,5....9). Thus each thread will be responsible from some lines and all lines will be red by all threads. Same algorithm works for all other threads (upper,replace)

**void \*read\_from\_file(void\* data):**

This function is for read threads. It reads from file and parse to a linked list.

After ech iteration replace and upper semaphore posted in order to upper and replace function work accordingly.

**void \*upper\_letter(void \*data) and void \*replace(void \* data):**

Both functions are responsible from thread functions.

**int calculate\_number\_of\_operations(int size,int id):**

This functions is for calculating how many times each thread will work as explained in structure part.

**void get\_number\_of\_line():**

Gets the number of line inside txt file in order to init a linked list with null values.

**void print\_list(data \*\* root):**

Prints all data inside array after all threads are done with thier task.

**void add\_node\_with\_data(data\*\*root,char \*line,int index):**

Since there is addition with null values in order to init linked list there is also init with data.

**void add\_node(data\*\*root):**

This function is for initializing linked list.

**void \*replace\_or\_upper(void \* flags):**

Honestly we have no idea why we add that function. We tried to use mutex lock function but it did not work as we expacted and it remained as it is.

**void replace\_letter(data \*\* root,int index):**

For replacing spaces with uncerscores

**void convert\_to\_upper(data \*\* root,int index):**

Same as above

**int is\_null(data \*\* root,int index):**

This function check if node of linked list with given index is wheter null or not.

Notes: There are some parts that they do not work.

1-)There is no write part.

2-)The upper and replace threads are working at the same time but they don work properly.

3-)When giving command line argument if number of upper threads are more than replace it stucks but when they are less than replace it does not stuck.

4-)We could not get file name as command line argument so please change file name with “deneme.txt”

Conclusion:We tried our best in order to achive this programming assigment but we could not make it. However we learned many thing.