Due Date: 11.01.2023 Wednesday, 11:59 PM

MARMARA UNIVERSITY

FACULTY OF ENGINEERING

COMPUTER ENGINEERING DEPARTMENT

CSE 3033 - OPERATING SYSTEMS PROJECT REPORT

Group Members:

Aimen Daddi 150119879

Nidanur Tekin 150119794

Bihter Nilüfer Akdem 150119810

Instructor:

Assoc. Prof. Ali Haydar Özer

Introduction

The primary goal of this project is to create a program that reads from a file, modifies its contents, then writes to the same file using sequences and synchronization. We have a program written on 4 different threads. These are read, upper, replace and write. The Read Thread reads lines at a time from a file and stores each line in an array. Upper Thread will read a string from the array, capitalize it, and then write it back to the same array index. The Replace Thread will read from the array, substitute the underscore character for each space, and then write back to the same array index. The array will be read by the Write Thread, which will then store each line in its proper location in the text file. We used the C programming language to create our scripts in the Linux operating system. For this software, a C built-in library called Pthread library was used.

How it works?

The Read Threads read from the text.txt file. Each Read Thread is assigned a different line number. In this case, a read thread will save line 0 of the text file at index 0 of the array if it reads it. The Upper Thread will scan the subject directories one at a time and capitalize the text after at least one item has been added to the topic. When the string has at least one entry, it will replace any spaces with underscores. All Replace Thread reads a different set of directories. We tried to make Replace Threads run simultaneously with Upper Threads. It's set to have only one Writer Thread that can write the file. We forced all threads to run concurrently in the system and made an effort to synchronize them. An attempt was made to create a queue between threads using semaphores. This queue was created for the first time the thread was to be used. A thread job checks to see if there is still work to be done after it has been completed. Worked to support numerous concurrent operations in various queue locations.