Farhan Hyder
CSC 332 – Assignment 2 summary
Submission date: 11/13/2018

Summary

For this project I have used multi-threading and mutex. I have chosen multi-threading over multi-processing for this project because as this is a relatively small project, choosing multi-threading seemed more economically beneficial. Mutex is used to ensure that only a single thread will have access to the main data file at a time.

Structs and functions that is created for this project are listed below.

Structs

- node_t: Represents a single node of a list. Each node has three fields – value (int), next (node), previous (node)
- list_t: Represents a list. Each list has two fields – head (node), tail (node). Tail node is used for faster insertion and faster merge.

Functions

+ list_t *read_file (char *filename): reads the given file, returns a list_t containing the numbers in the file.
+ void *thread_job(void *t): Core function of the project. Argument is the number of the file that the thread will use. First, reads the file and creates a list. Then, enters the critical region (lock mutex). Inside the critical region it reads the main data file and creates a list. The two lists are merged and the sorting algorithm starts. Algorithm is: during each iteration it finds the minimum number of the list. Then, the minimum number is written into the updated main data file and it's removed from the list. After this process is completed, program exits the critical region (unlock mutex) and makes the main data file available to others.
+ compare_lists(list_t *list1, list_t *list2): A supplementary function to check if two lists are equal or not. In this project, this is used to compare the updated main data file (datafile.dat) and the provided answer file by the instructor (ans.dat).

Instructions to compile and run the code

```
$ gcc -pthread main.c
$ ./a.out
```

possible output if run was successful: [EQUAL] Two lists are equal.