Gebze Technical University
Computer Engineering
CSE 344
System Programming
Final Project Report

SEDA KANIK
151044068

June 30, 2020

In this homework, we are responsible for manage the communication between client(s) and server. The main thread will read including of the file as graph and creates threads which gets some request from client(s) and answers them. The file contains edges which points from a vertex to another vertex.

My Solution:

There are 5 types of struct; for AdjacencyListNode, AdjacencyList, Graph, queue , cacheDS.
cacheDS: keeps the information about cache data structure, soruce, destination and path.
queue: keeps a queue array and indices of the rear and front.
Graph: keeps an array which type of AdjacencyList and visited array, and size of the array of the graph.
AdjacencyList: keeps a head which type of AdjacencyListNode.
AdjacencyListNode: keeps a destination and next which type of AdjacencyListNode.

Most of pointers and variables are created as global. Here is the work: First of all all parameter tests are done and setted some errno and printed screen then exited. Then reads all the file and creates graph respect to this points. File: I assumed the file is not empty and that its contents have the proper expected format. So, I do not need to check the content of the file.

1 Mutexes, Condition Variables and Purposes:

mutex_main: main thread use this mutex and sends a signal to additional thread and thread pool and waits some condition, thread pool use this mutex and sends a signal to additional thread and main thread
mutex_additional: thread pool use this mutex and increase the busy thread number and sends signal to additional thread to check and expand the number of thread and also sends signal to main to check and if the all threads are busy makes main to wait
mutex_m: used for cache, cache implemented as reader/writer paradigm like as lecture file
cond_thread: thread pool waits for some requests and if the request is accept, main thread sends a signal to thread pool
cond_additional: main sends signal to thread pool and additional thread to check if the number of busy thread is equal to created thread or not
cond_waiting: thread pool sends to main thread to check all the threads are busy or not to wait main to accept requests
cond_start: main thread accepts a signal and sends thread pool to starts answer to this request
okToRead: if there is no writer in cache database, reads cache database
okToWrite: if there is no reader and writer in cache database, writes cache database

2 Function Definitions:

2.1 pool:

Threads waits signal from main, if takes the signal, sends signal to additional to check percentage of busy threads and also sends to main to check

percentage of busy threads. Then reads source and destination from client, reads database and writes client and file if there is a path from source to destination, finds path with bfs and writes client and file and also database.
Bonus Part:
If user will not gives a value for r, initial value is -1 and assumes writer prioritization.
-r 0 : reader prioritization
Checks if there is active reader or writer in database while just before reading database, waits if there is some threads in database, then checks database if there is any path, if there is no path in database, it will find a path, writes it in database and answer it
-r 1 : writer prioritization
Checks if there is active or waiting writer in database while just before reading database, if there is no writer, reads database and it will find a path if there is no path in database, writes it in database and answer it
-r 2 : equal priorities to readers and writers
Lockes mutex and reads database, then unlock mutex, if there is no path in database, finds a path, writes it in database and answer it. It must not to wait while writing in database

2.2 timestamp:

Finds and prints time front of the every line of the file

2.3 check:

If the number is exists as vertex

2.4 numbersNodeEdge:

Calculates numbers of vertices and edges and keep vertex in arrays

2.5 createGraph:

creates a graph

2.6 addEdge:

adds an edge into graph

2.7 isEmpty:

check if the queue is empty or not

2.8 isAdjacent:

checks if the given vertices are adjacent or not

2.9 writeDatabase:

Writes database respect to –r, if the –r is not given, assumes writer prioritization

2.10 modifyQueue:

modifies the queue after BFS, checks from rear the queue to front looks for adjacency between two vertices

2.11 addQueue:

adding elements into queue

2.12 isSource:

checks if the vertex is source

2.13 isDestination:

checks if the vertex is destination

2.14 removeQueue:

Removing elements from queue

2.15 BFSandWriteDB:

visites vertices with BFS and adds queue, then modifies it

2.16 searchDatabase:

searches database for a taken path from client

2.17 createCache:

initalizes cache data structure

2.18 additional:

additional thread function

2.19 handler:

if the signal SIGINT reaches, keeps a flag as exit_signal and exits after the last thread execution