Gebze Technical University

Computer Engineering

CSE-344 System Programming

FINAL PROJECT REPORT

Student:
Nevra GÜRSES
161044071

1 INTRODUCTION

1.1 Project Definition

In this project , there must be developed 2 programs.A threaded server and a client. The server will load a graph from a text file, and it uses a dynamic pool of threads to handle incoming connections. The clients will connect to the server and request a path between two arbitrary nodes, and the server will provide this service.And also the server process will be a daemon.

2 METHOD

2.1 Problem Solution

For solving this problem,firstly I create daemon structure.After,I read file and I find node number.Because in some files of given link,there are different size of nodes from 3. line of input file.So,I guarantee that finding node number.After that, I write graph structure and I initialize this graph structure according to input file.Then I create socket structure.And I create threads according to given start number of threads in commandline argument.And I also create a dynamic pool thread.In main thread,I accept requests on socket and I add file descriptors of sockets in queue in mutexes and condition variables.Every thread has own mutexes and condition variables.I make adding queue operation in critical section.In server thread function,I take file descriptor number from queue and according to this file descriptor,I communicate with clients.I make communication in critical section in thread function.I control queue empty situation by condition variables of every thread.I read source and destination nodes from client by file descriptor of socket and I find path for client.Firstly I control cache structure.If path for source and destination is in cache structure,I take path from cache.If it is not in cache,I make BFS for finding path.I write results in client by socket.I use reader-writer paradigm for database.And I make reader-writer prioritization in thread functions.In dynamic pool function,I control whether thread pool is loading 75 % or not.If it is not 75 %,I wait by condition variable.If it reaches 75% I increase pool 25% and I create new threads.If sigint signal is entered,program is terminated.For client,I make communication via socket.

CACHE STRUCTURE OF SERVER:
For my cache structure,I use 2 struct data type.Fistly innerNode struct.In this struct,I keep path as char*,I keep length of path and I keep last node that is destination node of path.Other is Cache struct.In this struct, I use 2D pointer of innerNode structure.This is 2D pointer because in first dimention,it keeps every different source nodes.For example 0,1,2 etc.In second dimention it keeps paths as innerNode structure that are starting with node that is in first dimention.I use this 2 struct for cache structure because it has fast access/search and it has not duplicate elements.For example I collect paths that are starting with 0 in same first dimention.And second dimention I add paths as consecutive.If I show cache structure in my code:

```
32  /*Node for cache structure.*/
33  typedef struct innerNode {
34      int lastNum; //last number of path.
35      char* path; //path between 2 nodes.
36      int path_size; //size of path.
37  } innerNode;
38  /*Cache structure.*/
39  typedef struct cache {
40      innerNode** pathArr; //2D Innernode pointer to keep path between 2 node.
41      int pathArrLen; //keeps different source nodes.
42      int *innerNode_size; //keeps path sizes that are different source nodes.
43  } cache;
44  cache Cache;
```

DYNAMIC POOL OF SERVER:
For implement dynamic pool,I create a dynamic pool thread.In dynamic pool thread function I control whether thread number reached 75 % or not.If it did not reach,I wait condition variable signal.If signal is come,I increase pool 25 % and I reallocate thread area and I create new threads.I use mutex and condition variables for critical section control.

2.2 Functions that I use in Project for Server

- **int findNodeNum(FILE* fdInput):**This function finds total node number in file.
- **struct node* adjListNode(int dest):**This function creates adjacency list node.
- **struct Graph* initializeGraph():**This function initalizes graph for V vertex.

- **struct node* getNodeList(int src):**This function gets node list of given source.
- **int elementNum(int src):**This function finds connected node number of given source.
- **void addEdge(int src, int dest):**This function creates edge between 2 node that are source and destination.
- **int isEdge(int src, int destination):** This function controls whether there is an edge between given 2 node or not.
- **void printGraph():**This function prints graph structure.
- **void createGraph(FILE* fdInput):**This function creates graph structure according to reading input file.
- **void freeAllocatedGraph():**This function frees created graph structure.
- **void initializeCache():**This function initializes cache structure.
- **void freeCache():**This function frees cache structure.
- **void add_Cache(char* findedPath,int source,int dest):**This function adds path for given source and destination in cache structure.
- **char* getPath(int src,int dest):**This function gets path of given source and destination nodes from cache structure.
- **int controlCache(int source,int dest):**This function controls cache for finding whether given path is in cache or not.
- **struct queue* createQueue():** Creates and initalizes queue structure.
- **void addQueue(struct queue* Queue, int item):**Inserting element in queue.
- **void deleteQueue(struct queue* Queue) :**Deleting element from queue.

- **int controlEmpty():**This function for controlling whether queue is empty or not.
- **void display(struct queue* Queue)**This function for displaying queue elements.
- **int getFront(struct queue* Queue):**Getting first number from queue.
- **int getRear(struct queue* Queue):**Getting last number from queue.
- **int BFS(int src, int dest,int pred[], int dist[]) :**Breadth-first search function for finding path.This is helper function for finding path function.
- **char* findingPathWithBFS(int s,int dest) :**Finds shortest path between given 2 nodes.It applys BFS search algorithm for finding path.
- **void initilializeMutexesAndConds():**This function initializes thread mutexes and condition variables.
- **unsigned long getTime():**Getting time as microsecond.After using,this microsecond will converted second.
- **void func(int sockfd,int threadNum):**Function for comminication between client and server.Messages is sending to client in this function.And also reader-writer mechanism is used for informations of database structure.
- **static void *threadFunc(void *arg):**Thread function.
- **void *dynamic_control(void* arg):**Dynamic pool thread function.
- **void initializeQueue():**Initializing queue of file pointer of accept address of socket.
- **void freeQueue():**This function frees queue.
- **void daemon_code():**Daemeon code.
- **int main(int argc, char *argv[]):**Main function.

2.3 Functions that I use in Project for Client

- **unsigned long getTime():**Get time in micro second,while using in after,it will converted in second.
- **void func(int sockfd,int src,int dest,time_t startTime) :**This function to send and get informations with server.
- **int main(int argc, char *argv[]):**Main function.

NOTE: I also implement bonus part of project.

3 RUNNING RESULTS

After server operations,there is no memeory leak or error when SIGINT signal is come by kill command.If I show this,I run with valgrind:

```
nevra@ubuntu:~/Desktop$ gcc server.c -o server -Wall -Wextra -pedantic -g  -I . -pthread
nevra@ubuntu:~/Desktop$ valgrind --leak-check=full --show-leak-kinds=all ./server -i input.txt -p 8080 -o output.txt -s 4 -x 24 -r 2
==2970== Memcheck, a memory error detector
==2970== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==2970== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==2970== Command: ./server -i input.txt -p 8080 -o output.txt -s 4 -x 24 -r 2
==2970==
==2970==
==2970== HEAP SUMMARY:
==2970==     in use at exit: 0 bytes in 0 blocks
==2970==   total heap usage: 1 allocs, 1 frees, 552 bytes allocated
==2970==
==2970== All heap blocks were freed -- no leaks are possible
==2970==
==2970== For counts of detected and suppressed errors, rerun with: -v
==2970== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
nevra@ubuntu:~/Desktop$ ==2971==
==2971== HEAP SUMMARY:
==2971==     in use at exit: 0 bytes in 0 blocks
==2971==   total heap usage: 20,863 allocs, 20,863 frees, 2,170,916 bytes allocated
==2971==
==2971== All heap blocks were freed -- no leaks are possible
==2971==
==2971== For counts of detected and suppressed errors, rerun with: -v
==2971== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

nevra@ubuntu:~/Desktop$
```

Output example in log file accoring to given input file:

home > nevra > Desktop > output.txt

```
Executing with parameters:
-i input.txt
-p 8080
-o output.txt
-s 4
-x 24
-r 2
[ 1593538459.736187 sec]Loading graph...
[ 1593538460.132697 sec]Graph loaded in 0.325548 seconds with 6301 nodes and 20777 edges
[ 1593538460.349622 sec]Thread #0: waiting for connection
[ 1593538460.359381 sec]Thread #1: waiting for connection
[ 1593538460.560603 sec]Thread #2: waiting for connection
[ 1593538460.650335 sec]Thread #3: waiting for connection
[ 1593538460.651434 sec]A pool of 4 threads has been created
[ 1593538472.271834 sec]A connection has been delegated to thread id #0, system load 0.000000 %
[ 1593538472.293102 sec]Thread #0: no path in database, calculating 0->5555
[ 1593538482.087680 sec]Thread #0: path calculated:0->3->703->2635->2269->2500->3705->4892->5555
[ 1593538482.088724 sec]Thread #0: responding to client and adding path to database
[ 1593538483.502390 sec]A connection has been delegated to thread id #1, system load 0.000000 %
[ 1593538483.508712 sec]Thread #1: no path in database, calculating 225->876
[ 1593538483.866116 sec]Thread #1: path calculated:225->2178->31->637->876
[ 1593538483.866213 sec]Thread #1: responding to client and adding path to database
[ 1593538485.250737 sec]A connection has been delegated to thread id #2, system load 0.000000 %
[ 1593538485.258156 sec]Thread #2: searching database for a path from node 0 to node 5555
[ 1593538485.266054 sec]Thread #2:  path found in database:0->3->703->2635->2269->2500->3705->4892->5555
[ 1593538486.979251 sec]A connection has been delegated to thread id #3, system load 0.000000 %
[ 1593538486.994768 sec]Thread #3: no path in database, calculating 224->666
[ 1593538487.005477 sec]Thread #3: path not possible from node 224 to 666
[ 1593538498.053937 sec ] 2 Termination signal received, waiting for ongoing threads to complete.
[ 1593538498.056894 sec] thread is closing because SIGINT signal come.
[ 1593538498.068631 sec] thread is closing because SIGINT signal come.
[ 1593538498.070882 sec] thread is closing because SIGINT signal come.
[ 1593538498.072148 sec]Dynamic thread exiting because  SIGINT signal come.
[ 1593538498.072868 sec] thread is closing because SIGINT signal come.
```

[1c

After client operations,there is no memeory leak or error.If I show this,I run with valgrind: