CSE344 System Programming

Final Project Report

Ahmet DENİZLİ

161044020

1. Overview

In this project, There are 3 programs to be developed: servant, server and client. I did synchronization and communications beetween these programs with sockects and pthread methods.

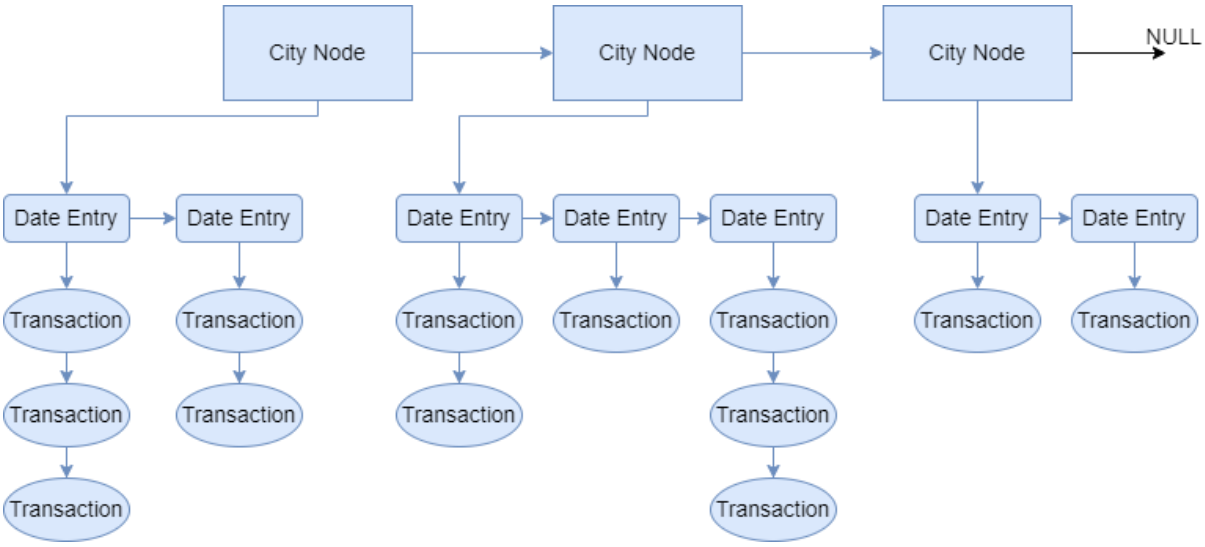
2. Design

For Server program;

Firstly initializes mutexes and creates handling threads. Opens the socket and accepts the connection. The main thread forwards incoming connections to the available thread in the pool. On each connection add connection file descriptor to the queue and sends condition signal to available handling thread. Main thread waits new connections until the sigint signal comes. When gets sigint signal, sends condition signal to all thread joins them, after they finished frees resources, destroys mutexes and close file descriptors before exit. If servant connected, gets informations and saves them to the new Servant struct object.

For Servant program;

I designed nested linked list for storing the dataset. Showed in the below figure.



Servant process, opens the given directory. Reads dataset and stores it to the nested linked lists. Data structure is that each city entry has date entry linked list, and each date entry has transaction linked list. So traversing and checking request is easy.

Finds its pid with checking each process in /proc. Opens socket and bind port “0” to get a unique port number. With getsockname function gets allocated port. Then connects to the server and sends its process id, responsible city names and port number. Then

listens its port until sigint occurs. When accept a new connection creates new servantThreadFun and detach it.

Servant threads gets the request, handle it and sends response to the client which is transaction count.

For Client program;

Opens the given file which is contains requests, reads the file. Then creates a thread for every request. Join threads to finish their job, threads will print on STDOUT. Then free resources and exits

For critical sections;

I used mutex so that the critical part is not a problem. For every queue operation locked mutex, maked call and then unlocked mutex.

For waiting with conditional variables;

In the server program, handling threads waits with condition variable. For each thread, I created a separate mutex to ensure that all threads can work in parallel. I created one condition variable for all threads. When a new connection come send condition signal and one of the available thread starts processing request.

3. Data Structure For the Dataset

```
struct cityNode {
    char name[50];
    struct dateNode *entries;
    struct cityNode *next;
};
```

```
struct dateNode {
    char rev_date[9];
    char date[11];
    struct dateNode *next;
    struct transactionNode *firstTransaction;
};
```

```
struct transactionNode {
    int t_id;
    char type[30];
    char street_name[30];
    int square_meter;
    int price;
    struct transactionNode *next;
};
```

I designed nested linked lists for the dataset. For each city sub-directory, created a new cityNode and added to linkedlist.

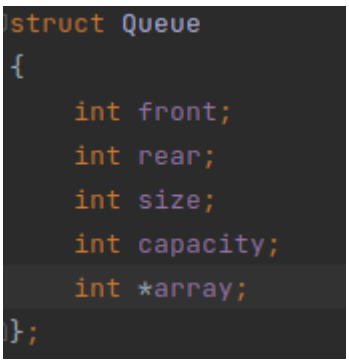
In each city-subdirectory there is a fixed number of ASCII files with a date as a filename in the format DD-MM-YYYY. I added rev_date attribute for dateNode struct, it is date but in the format YYYYMMDD. So for each file, created a new dateNode entry with its filename and reverse of date. Then added the

new node to the correct index by comparing rev_date attributes. This way datenode will be sorted. Finally added head of the dateNode linked list to the corresponding city.

Each date file's contents will start with a constant number of records as rows, and each record will represent a real-estate transaction that took place during that date at that city in the form of a space separated line. For each transaction, created a new transaction node and added it to the transaction linked list. Finally added head of the transaction linked list to the corresponding dateNode entry.

4. Queue Data Structure

For sending connection fds to the thread pool, I used queue data structure. It is simple queue also known as a linear queue. It is the most basic version of a queue. Insertion of an element, the Enqueue operation takes place at the rear end and removal of an element, the Dequeue operation takes place at the front end. When adding, if queue is full then reallocates array.



5. Signal Handling

Defined sigint handler function with sigaction for the server and the servant programs to exit. When takes SIGINT signal, programs change the global sigint flag.

The server process main thread stops listening port and sends condition signal to all handler threads and join them. Handler threads wake up and finish loop, then main thread free resources and exits.

The servant process main thread stops listening port, then free the dataset and resources, finally exits.

6. Functions

6.1. server.c

void sigint_handler(int signum): Handler for SIGINT signal for the server process. It sets sigint_flag to 1.

void *handlingThreadFun(void *arg): This function gets connection file descriptor from queue. When connection is from from servant process saves informations to the

new Servant struct object. When connection is from client, if request has city sends request to the related servant if not sends request to all servants. Finally sends response to the client.

int main(int argc, char *argv[]): Starts handling threads. Opens the socket and accepts the connection. The main thread forwards incoming connections to the available thread in the pool. On each connection add connection file descriptor to the queue and sends condition signal to available handling thread. Main thread waits new connections until the sigint signal comes.

6.2. servant.c

void sigint_handler(int signum): Handler for SIGINT signal for servant process. It sets sigint_flag to 1.

int check_proc(const char* proc_id, int cmd, int argc, char *argv[]): Controls given process id is the current process id, with cheking argv and given process cmdline.

void freeResources(): Function for freeing allocated resources like linked lists for the dataset, queue etc.

void *servantThreadFun(void *arg): In this function servant thread created from main thread. Gets server connection socket file descriptor from queue. Gets the request from server. Calculates transaction count for the given request. Sends result to the server with writing to the connection socket fd. Finally increase one total handled request variable and returns.

int main(int argc, char *argv[]): Opens the given directory. Reads dataset and stores it to the nested linked lists. Then connects to the server and sends its process id, responsible city names and port number. Then listens its port. When accept a new connection creates new servantThreadFun. When sigint occurs free resources and exits.

6.3. client.c

void *clientThreadFun(void *arg): This function takes ThreadParam struct as an argument. Waits for all threads to be created. Then opens sockect, connects to the server and sends the request. Finally receives the response and prints it to the STDOUT.

int main(int argc, char *argv[]): Opens the given file which is contains requests, reads the file. Then creates a thread for every request. Join threads to finish their job. Then free resources and exits.

6.4. LinkedList.h

struct dateNode* insertFirstDateLL(struct dateNode *head, char *rev_date, char *date, struct transactionNode *firstTransaction): This function creates new dateNode with given arguments. Then adds the new node to the first and returns new head of the date linkedlist. Revdate is date but in the format YYYYMMDD.

struct dateNode* sortedInsertDateLL(struct dateNode* sorted, char *rev_date, char *date, struct transactionNode *firstTransaction): This function creates new dateNode with given arguments. Then adds the new node to the correct index by comparing revdate parameters and returns head of the date linkedlist. Revdate is date but in the format YYYYMMDD.

struct transactionNode* insertFirstTransactionLL(struct transactionNode *head, int t_id, char *type, char *street_name, int square_meter, int price): This function creates new transactionNode with given arguments. Then adds the new transaction node to the first and returns new head of the transaction linkedlist.

void reverseTransactionLL(struct transactionNode head_ref):** This function reverses the given transaction linkedlist.

6.5. queue.h

struct Queue* create_queue(int capacity): This function creates a queue with given capacity. Creates queue in heap memory.

int isFull(struct Queue* queue): This function checks if queue is full or not. Returns 1 if queue size is equal to capacity, else returns 0.

int isEmpty(struct Queue* queue): This function checks if queue is empty or not. Returns 1 if queue size is zero, else returns 0.

int enqueue(struct Queue* queue, int item): This function adds given item to the queue. If queue is full, reallocates queue then adds the item. Returns the queue size.

int dequeue(struct Queue* queue): This function deletes front item from queue and returns it.

void free_queue(struct Queue* queue): This function frees allocated memory.

6.6. utilities.h

void errExit(char *s): This function prints given error via perror then exits.

#define BUF_SIZE 200: Defined buffer size as 200 bytes. It is same for all programs.

7. Sample Screenshots

Running Server

```
ubuntu@ubuntu:~/Desktop/344final$ ./server -p 33000 -t 5
[16/06/2022 06:50:41]Servant 92137 present at port 60773 handling cities ADANA-ARDAHAN
[16/06/2022 06:50:41]Servant 92142 present at port 55561 handling cities KASTAMONU-KUTAHYANU
[16/06/2022 06:50:41]Servant 92140 present at port 34505 handling cities EDIRNE-HAKKARI
[16/06/2022 06:50:41]Servant 92138 present at port 56899 handling cities ARTVIN-BITLIS
[16/06/2022 06:50:41]Servant 92143 present at port 58733 handling cities MALATYA-ORDUTYA
[16/06/2022 06:50:41]Servant 92144 present at port 52555 handling cities OSMANIYE-SIVASIYE
[16/06/2022 06:50:41]Servant 92139 present at port 55801 handling cities BOLU1-DUZCE
[16/06/2022 06:50:41]Servant 92145 present at port 34127 handling cities TEKIRDAG-ZONGULDAK
[16/06/2022 06:50:41]Servant 92141 present at port 33839 handling cities HATAY-KARSY
[16/06/2022 06:51:17]Request arrived "transactionCount TARLA 01-01-2073 30-12-2074 ADANA "
[16/06/2022 06:51:17]Contacting servant 92137
[16/06/2022 06:51:17]Request arrived "transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR"
[16/06/2022 06:51:17]Contacting servant 92138
[16/06/2022 06:51:17]Response received: 3, forwarded to client
[16/06/2022 06:51:17]Request arrived "transactionCount MERA 03-02-2018 09-11-2050 "
[16/06/2022 06:51:17]Contacting ALL servants
[16/06/2022 06:51:17]Request arrived "transactionCount BAHCE 02-03-2005 17-01-2084"
[16/06/2022 06:51:17]Contacting ALL servants
[16/06/2022 06:51:17]Request arrived "transactionCount DUKKAN 20-04-2000 23-01-2031  KILIS"
[16/06/2022 06:51:17]Contacting servant 92142
[16/06/2022 06:51:17]Request arrived "transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN"
[16/06/2022 06:51:17]Contacting servant 92137
[16/06/2022 06:51:17]Response received: 3, forwarded to client
[16/06/2022 06:51:17]Response received: 1, forwarded to client
[16/06/2022 06:51:17]Request arrived "transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA"
[16/06/2022 06:51:17]Contacting servant 92137
[16/06/2022 06:51:17]Response received: 5, forwarded to client
[16/06/2022 06:51:17]Response received: 4, forwarded to client
[16/06/2022 06:51:17]Request arrived "transactionCount VILLA 22-04-2049 20-03-2061"
[16/06/2022 06:51:17]Contacting ALL servants
[16/06/2022 06:51:17]Request arrived "transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA"
[16/06/2022 06:51:17]Contacting servant 92141
[16/06/2022 06:51:17]Request arrived "transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY"
[16/06/2022 06:51:17]Contacting servant 92137
[16/06/2022 06:51:17]Response received: 4, forwarded to client
[16/06/2022 06:51:17]Response received: 0, forwarded to client
[16/06/2022 06:51:17]Response received: 158, forwarded to client
[16/06/2022 06:51:17]Response received: 343, forwarded to client
[16/06/2022 06:51:17]Response received: 29, forwarded to client
^C[16/06/2022 06:51:32]SIGINT  has been received. I handled a total of 10 requests. Goodbye.
```

Running Servants

```
ubuntu@ubuntu:~/Desktop/344final$ ./script.sh
ubuntu@ubuntu:~/Desktop/344final$ Servant 92137: loaded dataset, cities ADANA-ARDAHAN
Servant 92137: listening at port 60773
Servant 92142: loaded dataset, cities KASTAMONU-KUTAHYA
Servant 92140: loaded dataset, cities EDIRNE-HAKKARI
Servant 92142: listening at port 55561
Servant 92140: listening at port 34505
Servant 92138: loaded dataset, cities ARTVIN-BITLIS
Servant 92138: listening at port 56899
Servant 92143: loaded dataset, cities MALATYA-ORDU
Servant 92143: listening at port 58733
Servant 92144: loaded dataset, cities OSMANIYE-SIVAS
Servant 92144: listening at port 52555
Servant 92139: loaded dataset, cities BOLU-DUZCE
Servant 92139: listening at port 55801
Servant 92141: loaded dataset, cities HATAY-KARS
Servant 92145: loaded dataset, cities TEKIRDAG-ZONGULDAK
Servant 92145: listening at port 34127
Servant 92141: listening at port 33839
Servant 92141: termination message received, handled 4 requests in total.
Servant 92144: termination message received, handled 3 requests in total.
Servant 92145: termination message received, handled 3 requests in total.
Servant 92139: termination message received, handled 3 requests in total.
Servant 92138: termination message received, handled 4 requests in total.
Servant 92143: termination message received, handled 3 requests in total.
Servant 92142: termination message received, handled 4 requests in total.
Servant 92140: termination message received, handled 3 requests in total.
Servant 92137: termination message received, handled 7 requests in total.
```

Running Client

```
ubuntu@ubuntu:~/Desktop/344final$ ./client -r ./requestFile -q 33000 -s 127.0.0.1
Client: I have loaded 10 requests and I'm creating 10 threads.
Client-Thread-0: Thread-0 has been created
Client-Thread-1: Thread-1 has been created
Client-Thread-4: Thread-4 has been created
Client-Thread-5: Thread-5 has been created
Client-Thread-6: Thread-6 has been created
Client-Thread-7: Thread-7 has been created
Client-Thread-8: Thread-8 has been created
Client-Thread-9: Thread-9 has been created
Client-Thread-3: Thread-3 has been created
Client-Thread-2: Thread-2 has been created
Client-Thread-1: I am requesting "transactionCount MERA 03-02-2018 09-11-2050 "
Client-Thread-0: I am requesting "transactionCount TARLA 01-01-2073 30-12-2074 ADANA "
Client-Thread-4: I am requesting "transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR"
Client-Thread-3: I am requesting "transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN"
Client-Thread-5: I am requesting "transactionCount BAHCE 02-03-2005 17-01-2084"
Client-Thread-2: I am requesting "transactionCount DUKKAN 20-04-2000 23-01-2031  KILIS"
Client-Thread-6: I am requesting "transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA"
Client-Thread-7: I am requesting "transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY"
Client-Thread-8: I am requesting "transactionCount VILLA 22-04-2049 20-03-2061"
Client-Thread-9: I am requesting "transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA"
Client-Thread-0: The server's response to "transactionCount TARLA 01-01-2073 30-12-2074 ADANA " is 3
Client-Thread-0: Terminating
Client-Thread-4: The server's response to "transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR" is 3
Client-Thread-4: Terminating
Client-Thread-2: The server's response to "transactionCount DUKKAN 20-04-2000 23-01-2031  KILIS" is 1
Client-Thread-2: Terminating
Client-Thread-7: The server's response to "transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY" is 0
Client-Thread-7: Terminating
Client-Thread-6: The server's response to "transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA" is 4
Client-Thread-6: Terminating
Client-Thread-9: The server's response to "transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA" is 4
Client-Thread-9: Terminating
Client-Thread-3: The server's response to "transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN" is 5
Client-Thread-3: Terminating
Client-Thread-1: The server's response to "transactionCount MERA 03-02-2018 09-11-2050 " is 158
Client-Thread-1: Terminating
Client-Thread-5: The server's response to "transactionCount BAHCE 02-03-2005 17-01-2084" is 343
Client-Thread-5: Terminating
Client-Thread-8: The server's response to "transactionCount VILLA 22-04-2049 20-03-2061" is 29
Client-Thread-8: Terminating
Client: All threads have terminated, goodbye.
ubuntu@ubuntu:~/Desktop/344final$
```