GEBZE TECHNICAL UNIVERSITY

CSE344
SYSTEM PROGRAMMING COURSE

FINAL PROJECT REPORT

BARAN SOLMAZ
1801042601

Problem Defination:

There are 3 programs to be developed:

Servants: Reads datasets and store it in data structure,
Notify server for their dataset interval and port number,
Waits for requests from server.

Servers: Creates threads for incoming connections,
Waits for clients' requests and sends to servants,
Waits for servants' responds and sends to client.

Client: Reads request file and creates threads as many as request number,
Waits for other threads to send request,
Connects server and sends request,
Waits for responds.

Client:

```
./client -r requestFile -q PORT -s IP
```

Reads request file and create threads.
Sends request to server over given PORT and IP.

```c
void *client(void *input){
    char *req;
    req = (char *)input;
    char *token = strtok_r(req, ":", &req);
    int id=atoi(token);
    printf("%sClient-Thread-%d: Thread-%d has been created\n",getTimeStamp(),id,id);
    pthread_mutex_lock(&mutex);
    arrived++;
    while (arrived < requestCount){
        if (sig_check_thread() == 1)
            break;
        pthread_cond_wait(&cond, &mutex);
        if (sig_check_thread() == 1)
            break;
    }
    pthread_cond_broadcast(&cond);
    pthread_mutex_unlock(&mutex);
    printf("%sClient-Thread-%d: I am requesting "/%s".\n",getTimeStamp(),id,req);

    int result=socketOperations(req);
    printf("%sClient-Thread-%d: The server's response to "/%s" is %d.\n",getTimeStamp(),id,req,result);
    printf("%sClient-Thread-%d: Terminating.\n",getTimeStamp(), id);
    return NULL;
```

Waits for other threads.

Sends request to server and gets respond

Server:

Gets notification from servants and requests from clients.

```
./server -p PORT -t numberOfThreads
```

Server Threads:

```c
void *server(void *input){
    while (1){
        char *buffer = (char *)malloc(sizeof(char) * 1024);
        char *head=buffer;
        pthread_mutex_lock(&mutex);
        while (1 > queue->size){
            if (sig_check_thread() == 1)
                break;
            pthread_cond_wait(&cond, &mutex);
            if (sig_check_thread() == 1)
                break;
        }
        int socket_fd=dequeue(queue);
        pthread_mutex_unlock(&mutex);
        if (sig_check_thread() == 1)
            break;
        memset(buffer, '\0', 1024);

        read(socket_fd, buffer, 1024);
        if (sig_check_thread() == 1)
            break;
        if (buffer[0] == 'c'){
            buffer += 2;
            clientRequest(socket_fd,buffer);
        }else if (buffer[0] == 'n'){
            buffer += 2;
            addServant(buffer);
        }
        close(socket_fd);
        free(head);
        if (sig_check_thread() == 1)
            break;
    }
    return NULL;
}
```

Wait for the Queue size

Pops first element

Read from socket

if read line is from client ,send to servants

if read line is from servant,add servant

n for notify server about servant
c for client request

Main Server Thread:

```
while (1){
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,(socklen_t *)&addrlen)) < 0)
        break;
    if (sig_check_thread() == 1)
        break;

    pthread_mutex_lock(&mutex);
    enqueue(queue,new_socket);
    pthread_mutex_unlock(&mutex);
    pthread_cond_signal(&cond);
    if (sig_check_thread() == 1)
        break;
}
pthread_cond_broadcast(&cond);
joinThreads(server_threads);
for (int i = 0; i < servantCount; i++)
    kill(servants[i].id,SIGINT);
shutdown(server_fd, SHUT_RDWR);
```

Accepts connections adds thier socket ids to queue

Then wakes up one of the server threads.

Servant:

Reads folders and stores in AVL-Tree

```
./servant -d directoryPath -c 10-19 -r IP -p PORT
```

```
procID = getProcID("/proc/self/stat");
getFolders();

printf("Servant %d: loaded %s, cities %s-%s\n", procID, directoryPath,startDir_name,endDir_name);

int servant_fd = -1,err=-1;
sig_check();
for (int i = 0; i <200; i++){
    err= findEmptyPort(&servant_fd,servantPort);
    if(sig_check_thread()==1)
        break;
    if (err>=0)
        break;
    servantPort++;
}
printf("Servant %d: listening at port %d\n",procID,servantPort);
sig_check();
notifyServer(servant_fd,startDir_name, endDir_name, servantPort);
sig_check();
socketOperations(servant_fd);
freeNodeCity(root);
printf("Servant %d: termination message received, handled %d requests in total.\n",procID,requestCount);
return 0;
```

To get pid

To find empty port

Notify server about port and cities

Waits for requests from server

Data Structures:

Queue:

```
struct Queue
{
    int front, rear, size;
    int capacity;
    int *array;
};


int dequeue(struct Queue *queue);            // To get first element
int front(struct Queue *queue);              // Function to get front of queue
int rear(struct Queue *queue);               // Function to get rear of queue
int isEmpty(struct Queue *queue);              // Checks queue if it empty return 1
int isFull(struct Queue *queue);               // Checks queue if it full return 1
struct Queue *createQueue();                   // Constructor
void enqueue(struct Queue *queue, int item); // To insert element
```

AVL-Tree :

```
struct TransactionNode{
    char street[30];
    char type[20];
    int area;
    int height;
    int id;
    int price;
    struct TransactionNode *leftNode;
    struct TransactionNode *rightNode;
};
```

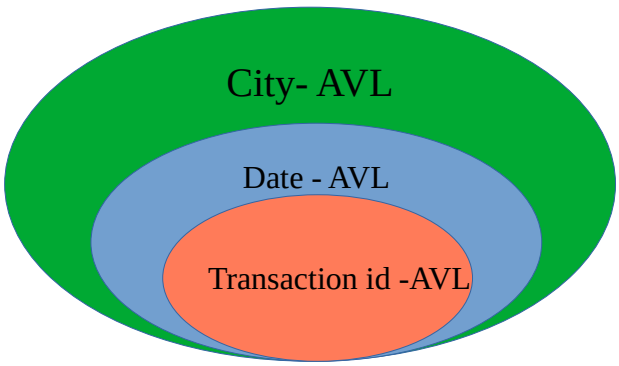
Balanced by transaction id

```
struct DateNode{
    char date[11];
    struct TransactionNode *transactions;
    struct DateNode *leftNode;
    struct DateNode *rightNode;
    int height;
};
```

Balanced by file name / date

```
struct CityNode{
    char city[30];
    int height;
    struct CityNode *leftNode;
    struct CityNode *rightNode;
    struct DateNode *dates;
};
```

Balanced by folder name / city



AVL Tree Functions:

Transaction:

```
int getBalanceTransaction(struct TransactionNode *N);                                    // Get the balance factor
int getID(char *key);                                                                    // Returns Transaction ID
int heightTransaction(struct TransactionNode *N);                                        // Calculate height
int max(int a, int b);                                                                   // Returns Max number
int searchTransaction(struct TransactionNode *root, struct Request *req);                // Search Transaction
struct TransactionNode *insertNodeTransaction(struct TransactionNode *root, char *key); // Insert node
struct TransactionNode *leftRotateTransaction(struct TransactionNode *x);                // Left rotate
struct TransactionNode *minValueTransactionNode(struct TransactionNode *node);           // Returns min value Node
struct TransactionNode *newTransactionNode(char *line);                                  // Create a node
struct TransactionNode *rightRotateTransaction(struct TransactionNode *y);               // Right rotate
void freeNodeTransaction(struct TransactionNode *root);                                  // Free Transactions
void printPreOrderTransaction(struct TransactionNode *root);                             // Print the tree
```

Date:

```
int cmpDate(char *key, char *date);                                          // Compares Dates
int getBalanceDate(struct DateNode *N);                                      // Get the balance factor
int getDate(char *date, int in);                                             // Returns DD/MM/YYYY -> in= 0/1/2
int heightDate(struct DateNode *N);                                          // Calculate height
int searchDate(struct DateNode *root, struct Request *req);                  // Search Date
struct DateNode *insertNodeDate(struct DateNode *root, struct DateNode *node); // Insert node
struct DateNode *leftRotateDate(struct DateNode *x);                         // Left rotate
struct DateNode *minValueDateNode(struct DateNode *node);                    // Returns min value Node
struct DateNode *newDateNode(char *line);                                    // Create a node
struct DateNode *rightRotateDate(struct DateNode *y);                        // Right rotate
void freeNodeDate(struct DateNode *root);                                    // Free Dates
void printPreOrderDate(struct DateNode *root);                               // Print the tree
```

City:

```
int getBalanceCity(struct CityNode *N); // Get the balance factor
int heightCity(struct CityNode *N);     // Calculate height
int searchCity(struct CityNode *root, struct Request *req); // Search City
struct CityNode *insertNodeCity(struct CityNode *root, struct CityNode *node); // Insert node
struct CityNode *leftRotateCity(struct CityNode *x);                           // Left rotate
struct CityNode *minValueCityNode(struct CityNode *node);                      // Returns min value Node
struct CityNode *newCityNode(char *city);                                      // Create a node
struct CityNode *rightRotateCity(struct CityNode *y);                          // Right rotate
void freeNodeCity(struct CityNode *root);                                      // Free Citys
void printPreOrderCity(struct CityNode *root);                                 // Print the tree
```

Functions:

Client:

```
void *client(void *in);//client process
void checkArgc(int argc, char *argv[]);//Arg check
void createThreads(char arr[][80], pthread_t *threads);
void handler(int sig_number);
void initialize_mutex();
void joinThreads(pthread_t *threads);
void readFile();//To read given file
void remove_mutex();
void sig_check();
int sig_check_thread();
int socketOperations(char *req);//Socket op. for client
```

Servant:

```
int findEmptyPort(int *servant_fd, int port);//to find empty port
int sig_check_thread();//thread sigint check
struct Request *parseRequest(char *req);//To parse incoming line
struct TransactionNode *readFiles(char *dir, char *fileName);//To read file in directory
void *servant(void *in);//servant process
void checkArgc(int argc, char *argv[]);//arg check
void freeReq(struct Request *req);//to free requests
void getFolders();//to get folders in directory
void handler(int sig_number);
void notifyServer(int servant_fd,char *start, char *end, int port_num);//To notify server
void readFolders(char folderNames[][30]);//to read folders with files
void sig_check();
void socketOperations();//Socket op. for servant
void sortFolders(char folderNames[][30], int size);//To sort folder name
```

Server:

```
char *getCity(char *req, int *index); //Returns city or null
int sendReqToServant(char *servant_ip, int servant_port, char *req);//sends request to servant
int sendReqToServants(char *req);   //To send req to all servants
int sendToServant(char *req); // sends to servant
int sig_check_thread(); //checks SIGINT in thread
int socketOperations();//socket op.
struct sockaddr_in *createSocket(struct sockaddr_in *address, int *server_fd);//To create socket
void *server(void *input);//server thread process
void addServant(char *buffer);//adds servant to array
void checkArgc(int argc, char *argv[]);//checks arg.
void clientRequest(int socket_fd, char *buffer);//To handle client req
void createThreads(int *arr, pthread_t *threads);
void freeQueue();//free Queue
void handler(int sig_number);
void initialize_mutex();
void joinThreads(pthread_t *threads);
void remove_mutex();
void sendResToClient(int sd, char *req);//send response to client
void sig_check();//main thread sigint check
```

TEST:

---script.sh

```
#!/bin/bash

PORT=33000

./server -p $PORT -t 11 &
sleep 1
./servant -d dataset -c 1-9 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 10-18 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 19-27 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 28-36 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 37-45 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 46-54 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 55-63 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 64-72 -r 127.0.0.1 -p $PORT &
./servant -d dataset -c 73-81 -r 127.0.0.1 -p $PORT &
sleep 3
./client -r requestFile -q $PORT -s 127.0.0.1
```

Output:

```
Servant 530157: loaded dataset, cities EDIRNE-HAKKARI
Servant 530157: listening at port 16000
Thu Jun 16 08:25:11 2022 Servant 530157, present at port 16000, handling cities EDIRNE - HAKKARI
Servant 530160: loaded dataset, cities MALATYA-ORDU
Servant 530160: listening at port 16001
Thu Jun 16 08:25:11 2022 Servant 530160, present at port 16001, handling cities MALATYA - ORDU
Servant 530155: loaded dataset, cities ARTVIN-BITLIS
Servant 530155: listening at port 16002
Thu Jun 16 08:25:11 2022 Servant 530155, present at port 16002, handling cities ARTVIN - BITLIS
Servant 530159: loaded dataset, cities KASTAMONU-KUTAHYA
Servant 530159: listening at port 16003
Thu Jun 16 08:25:11 2022 Servant 530159, present at port 16003, handling cities KASTAMONU - KUTAHYA
Servant 530154: loaded dataset, cities ADANA-ARDAHAN
Servant 530154: listening at port 16004
Thu Jun 16 08:25:11 2022 Servant 530154, present at port 16004, handling cities ADANA - ARDAHAN
Servant 530158: loaded dataset, cities HATAY-KARS
Servant 530158: listening at port 16005
Thu Jun 16 08:25:11 2022 Servant 530158, present at port 16005, handling cities HATAY - KARS
Servant 530162: loaded dataset, cities TEKIRDAG-ZONGULDAK
Servant 530162: listening at port 16006
Thu Jun 16 08:25:11 2022 Servant 530162, present at port 16006, handling cities TEKIRDAG - ZONGULDAK
Servant 530161: loaded dataset, cities OSMANIYE-SIVAS
Servant 530161: listening at port 16007
Thu Jun 16 08:25:11 2022 Servant 530161, present at port 16007, handling cities OSMANIYE - SIVAS
Servant 530156: loaded dataset, cities BOLU-DUZCE
Servant 530156: listening at port 16008
Thu Jun 16 08:25:11 2022 Servant 530156, present at port 16008, handling cities BOLU - DUZCE
```

```
Thu Jun 16 08:25:14 2022  Client: I have loaded 10 requests and I'm creating 10 threads.
Thu Jun 16 08:25:14 2022 Client-Thread-0: Thread-0 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-1: Thread-1 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-2: Thread-2 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-3: Thread-3 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-4: Thread-4 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-5: Thread-5 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-6: Thread-6 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-7: Thread-7 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-8: Thread-8 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-9: Thread-9 has been created
Thu Jun 16 08:25:14 2022 Client-Thread-9: I am requesting “/transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA”.
Thu Jun 16 08:25:14 2022 Client-Thread-0: I am requesting “/transactionCount TARLA 01-01-2073 30-12-2074 ADANA”.
Thu Jun 16 08:25:14 2022 Client-Thread-1: I am requesting “/transactionCount MERA 03-02-2018 09-11-2050”.
Thu Jun 16 08:25:14 2022 Client-Thread-2: I am requesting “/transactionCount DUKKAN 20-04-2000 23-01-2031 KILIS”.
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA”
Thu Jun 16 08:25:14 2022 Contacting Servant 530158
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount TARLA 01-01-2073 30-12-2074 ADANA”
Thu Jun 16 08:25:14 2022 Client-Thread-4: I am requesting “/transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR”.
Thu Jun 16 08:25:14 2022 Contacting Servant 530154
Thu Jun 16 08:25:14 2022 Client-Thread-6: I am requesting “/transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA”.
Thu Jun 16 08:25:14 2022 Client-Thread-3: I am requesting “/transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN”.
Thu Jun 16 08:25:14 2022 Client-Thread-5: I am requesting “/transactionCount BAHCE 02-03-2005 17-01-2084”.
Thu Jun 16 08:25:14 2022 Client-Thread-8: I am requesting “/transactionCount VILLA 22-04-2049 20-03-2061”.
Thu Jun 16 08:25:14 2022 Client-Thread-7: I am requesting “/transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY”.
```

```
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount MERA 03-02-2018 09-11-2050”
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR”
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA”
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount DUKKAN 20-04-2000 23-01-2031 KILIS”
Thu Jun 16 08:25:14 2022 Contacting Servant 530159
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount BAHCE 02-03-2005 17-01-2084”
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount VILLA 22-04-2049 20-03-2061”
Thu Jun 16 08:25:14 2022 Contacting All Servants
Thu Jun 16 08:25:14 2022 Contacting All Servants
Thu Jun 16 08:25:14 2022 Contacting All Servants
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN”
Thu Jun 16 08:25:14 2022 Contacting Servant 530154
Thu Jun 16 08:25:14 2022 Contacting Servant 530154
Thu Jun 16 08:25:14 2022 Contacting Servant 530155
Thu Jun 16 08:25:14 2022 Request arrived “transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY”
Thu Jun 16 08:25:14 2022 Contacting Servant 530154
Thu Jun 16 08:25:14 2022 Response received: 4, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-9: The server's response to “/transactionCount IMALATHANE 04-06-2004 11-11-2011 ISPARTA” is 4.
Thu Jun 16 08:25:14 2022 Client-Thread-9: Terminating.
Thu Jun 16 08:25:14 2022 Response received: 16, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-2: The server's response to “/transactionCount DUKKAN 20-04-2000 23-01-2031 KILIS” is 16.
Thu Jun 16 08:25:14 2022 Response received: 4, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-2: Terminating.
Thu Jun 16 08:25:14 2022 Client-Thread-7: The server's response to “/transactionCount AMBAR 28-01-2044 13-09-2050 AKSARAY” is 4.
Thu Jun 16 08:25:14 2022 Client-Thread-7: Terminating.
Thu Jun 16 08:25:14 2022 Response received: 52, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-3: The server's response to “/transactionCount BAG 01-12-2004 27-09-2089 ADIYAMAN” is 52.
Thu Jun 16 08:25:14 2022 Client-Thread-3: Terminating.
```

```
Thu Jun 16 08:25:14 2022 Response received: 39, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-4: The server's response to “/transactionCount FIDANLIK 02-09-2016 12-09-2081 BALIKESIR” is 39.
Thu Jun 16 08:25:14 2022 Client-Thread-4: Terminating.
Thu Jun 16 08:25:14 2022 Response received: 37, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-6: The server's response to “/transactionCount FABRIKA 22-07-2004 11-05-2072 ANKARA” is 37.
Thu Jun 16 08:25:14 2022 Client-Thread-6: Terminating.
Thu Jun 16 08:25:14 2022 Response received: 3, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-0: The server's response to “/transactionCount TARLA 01-01-2073 30-12-2074 ADANA” is 3.
Thu Jun 16 08:25:14 2022 Client-Thread-0: Terminating.
Thu Jun 16 08:25:14 2022 Response received: 29, forwarded to client
Thu Jun 16 08:25:14 2022 Response received: 158, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-1: The server's response to “/transactionCount MERA 03-02-2018 09-11-2050” is 158.
Thu Jun 16 08:25:14 2022 Client-Thread-1: Terminating.
Thu Jun 16 08:25:14 2022 Client-Thread-8: The server's response to “/transactionCount VILLA 22-04-2049 20-03-2061” is 29.
Thu Jun 16 08:25:14 2022 Client-Thread-8: Terminating.
Thu Jun 16 08:25:14 2022 Response received: 343, forwarded to client
Thu Jun 16 08:25:14 2022 Client-Thread-5: The server's response to “/transactionCount BAHCE 02-03-2005 17-01-2084” is 343.
Thu Jun 16 08:25:14 2022 Client-Thread-5: Terminating.
Thu Jun 16 08:25:14 2022 Client : All threads have terminated,goodbye.
```

```
baran@Linux:~/Desktop/1801042601$ kill -2 530130
baran@Linux:~/Desktop/1801042601$ Thu Jun 16 08:27:49 2022 SIGINT has been received. I handled a total of 10 requests. Goodbye.
Servant 530157: termination message received, handled 3 requests in total.
Servant 530160: termination message received, handled 3 requests in total.
Servant 530158: termination message received, handled 4 requests in total.
Servant 530159: termination message received, handled 4 requests in total.
Servant 530161: termination message received, handled 3 requests in total.
Servant 530162: termination message received, handled 3 requests in total.
Servant 530156: termination message received, handled 3 requests in total.
Servant 530154: termination message received, handled 7 requests in total.
Servant 530155: termination message received, handled 4 requests in total.

baran@Linux:~/Desktop/1801042601$
```