Nikolas Iliopoulos 1115201800332

3 Exercise System Programming

Compile & Run

Makefile is included. Use the command **make** to compile the source code.

To run the application use: **./travelMonitorClient -m numMonitor -b socketBufferSize -c cyclicBufferSize -s sizeOfBloom -i input_dir/ -t numThreads**

> *Note* that the order of the arguments doesn't matter.

Everything is the same with exercise 2 but instead of pipes there are networking sockets and the servers process the records with threads.

Structure & Classes

I used all the structures from the exercise 1 & 2 that i made

travelMonitor

```
class travelMonitorClient {
public:
    // Methods
private:
    struct hostent* ip;
    struct in_addr** ip_addr;

    struct sockaddr_in client;
    struct sockaddr* clientptr;

    char machineName[256];
    char externalAddress[256];

    string* command;
    int numMonitors;
    int socketBufferSize;
    int cyclicBufferSize;
    int sizeOfBloom;
    string input_dir;
    int numThreads;

    monitorList* monitors;
    monitorCountryPairList* countryToMonitor;

    stringList* viruses;
    statsList* requests;
    stringList* filePaths;
    stringList* countries;
    bloomFilterList* blooms;
};
```

MonitorServer

```
class monitorServer {
public:
    // Methods
private:
    struct hostent* ip;
    struct in_addr** ip_addr;

    struct sockaddr_in server;
    struct sockaddr* serverptr;

    char machineName[256];
    char externalAddress[256];

    pthread_mutex_t mutex;
    cyclicBuffer* buff;
    pthread_t* threads;

    int id;
    int port;
    int sock;
    int numThreads;
    string* command;
    int t;
    int f;
    string generalDirectory;

    char** argPaths;
    int argNumPaths;

    int socketBufferSize;
    int cyclicBufferSize;
    int bloomSize;

    treeNode* tree;
    bloomFilterList* blooms;
    skipList_List* skipLists;
    stringList* viruses;
    stringList* countries;
    stringList* filesReaded;
};
```

cyclicBuffer

I took the implementation from the lectures.

```
class cyclicBuffer {
public:
    string take();
    void put(string txt);
private:
    string* buff; // the buffer where the txt will be hold
    int start; // the position of the first txt
    int end; // the position of the last txt
    int count;// how many txts are in the buffer
    int size; // the size of the buffer
    int txtNumber; // the total txt that the travelMonitorCilent wants to insert
    int txtParsed; // how many txt are fully parsed

    pthread_mutex_t mtx; // the mutex that is used in the take and put
    pthread_cond_t cond_nonempty;// condition variable that stops the put() to
insert a txt if there is no space
    pthread_cond_t cond_nonfull;// condition variable that stops the take() to
remove a txt if there is no txt in the buffer
};
```

monitorCountryPairList

This list has the information about which country is assigned to which monitor.

```
class monitorCountryPairList
{
public:
    // Methods
private:
    string country;
    int monitor;
    monitorCountryPairList* next;
};
```

monitorList

This list has the read and write file decriptor for each monitor. As well as the pid and unique id number that i use.

```
class monitorList
{
public:
    // Methods
private:
    int readFD;
    int writeFD;
    int pid;
    int id;

    monitorList* next;
};
```

statsList

This list holds has ont node for every request. So i can make the statistics in the /travelStats command.

```
class statsList
{
public:
    // Methods
private:
    string country;
    string virusName;
    date d;
    bool stat;
    statsList* next;
};
```

Commands

/travelRequest

Must have exactly 5 arguments. At the start i check:

- id is only numbers
- 0 <= id <= 9999

If the virus is not in the virus list it is obvious we dont have to look the bloom filter to see that this id has not made the virus so return **REQUEST REJECTED – YOU ARE NOT VACCINATED**. If we have that virus we hash the id and check the 16 bits to be set to 1. If all of them are 1 then we send the command to the correct Monitor to take an answer. Then if the citizen has done the virus we check if it has passed 6 months.

I store every request in a list so i can make the statistics in the second command.

/travelStats

I check every request that i have in the request list and abjust two counts accordingly.

/addVaccinationRecords

Just send the command to the correct monitor. The monitor cathes it and loops throw all the files and put the not processed files to the cyclic buffer in the same way as before. Then the threads take the files and update the structures. Then all the bloomfilters are send back to the travelMonitorClient.

/searchVaccinationStatus

- Send to the Monitors the command
- if the monitor does not have the citizen send me -1 if it has returns 1.
- Then i read from the monitor that sends me 1 and print the things that it sends me.