Nikolas Iliopoulos 1115201800332

2 Exercise System Programming

Compile & Run

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

To run the application use: **./travelMonitor -m inputFile.txt -b bufferSize -s sizeOfBloom -i input_dir**

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

Error Handling & Checking

- Syntax of Records
    - The line must have 7 or 8 words
    - id is only numbers
    - 0 <= id <= 9999
    - age is only numbers
    - 1 <= age <= 120
    - 7 word must be either YES or NO
    - If there are 7 words total the seventh word must be NO
    - If there are 8 words total the seventh word must be YES
    - The date if exists must have the format dd-mm-yyyy
    - 1 <= dd <= 30 & 1 <= mm <= 12 & 1950 <= yyyy <= 2050
- Bad dupicates
    - Citizens with same id must match the credentials firstname lastname country age in order to be a valid duplicate of that citizen
    - Citizens that have valid credentials must also have a different virus name that the already inserted citizen has.

General

- The first thing i send to the Monitor is some usefull things such us his unique id, bufferSize, sizeofbloom, input_dir.
- I have functions sendStr() and receiveStr(). That reads and writes ftom the correct pipe.
- At the start of sending a string from the pipe. I firstly send the length of the string and then the string in parts based on the bufferSize. Then the receiver will know the length of the string that will come and in how many parts it will be send. Then it takes those strings parts and concatenate them.
- In order to know when i have to stop reading strings i made the convension when the length of the string is -1 it means i have to stop.

- To catch the signals i used sigaction()

Structure & Classes

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

travelMonitor

```
class travelMonitor {
public:
    // Methods
private:
    // the arguments
    int numMonitors;
    int bufferSize;
    int sizeOfBloom;
    string input_dir;

    struct sigaction handlerSIGINT_SIGQUIT; // to catch SIGINT & SIGQUIT
    struct sigaction handlerSIGCHLD; // to catch SIGCHLD
    string* command; // used for storing the command from the cin

    monitorList* monitors; // a list that is storing information
    // about each monitor [readFD, writeFD, pid, id]

    monitorCountryPairList* countryToMonitor; // a list that is storing
    // the countries of each monitor

    stringList* viruses; // ALL the viruses
    stringList* countries; // ALL the countries
    bloomFilterList* blooms; // ALL the blooms
    statsList* requests; // i store every request here
    // so i can make the statistics
};
```

Monitor

```
class Monitor {
public:
    // Methods
private:
    int id; // the unique id of each monitor
    // this is the first thing that they will receive from the pipe

    // the Arguments
    int bufferSize;
    int bloomSize;
    string generalDirectory;
```

```
    struct sigaction handlerSIGINT_SIGQUIT;  // to catch SIGINT & SIGQUIT
    struct sigaction handlerSIGUSR1; // to catch SIGUSR1

    int t; // a count of the accepted
    int f; // a count of the rejected

    // for named pipes FIFOs
    string readFifo;
    int readFD;
    string writeFifo;
    int writeFD;

    string* command; // used for storing the command from the cin

    treeNode* tree; //  a binary tree of the citizens
    bloomFilterList* blooms; // a list of the blooms
    skipList_List* skipLists; // a list of the 2 skiplists
    stringList* viruses; // a lsit of viruses
    stringList* countries; // a list of countries
    stringList* filesReaded; // a list of the files that are already processed
};
```

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.

/travelStats

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

/addVaccinationRecords

I send a signal SIGUSR1 to the correct Monitor. The monitor cathes it and loops throw all the files. And if the file is not in the files list it means that it is a new file. So i update that structures with this file and continue searching for other files.

/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.

Script

The script is in the **script\create_infiles.sh**.

The script uses array to store the countries. At first i create the directories with the all files. Then i start from the start of the inputFile and for each record i go to the spesific directory and put the record. In order to know in which txt i have to put the record i have an associated array and for each country i store and int that corresponds to the txt that the next record will go.