HELLENIC REPUBLIC
National and Kapodistrian University of Athens
EST. 1837

DEPARTMENT OF
INFORMATICS & TELECOMMUNICATIONS

Disease Aggregator

Anna-Aikaterini Kavvada - 1115201500050

June 3, 2020

Professor: Alexandros Ntoulas

1 Introduction

The project has been implemented in C language, using the jetbrain's editor Clion and tested both at my local linux and the departments linux machines, using ssh protocol for the connection.
Inside the tar.gz file the project is organised as shown below

- header: all the header files associated with the project
- src: all the .c files associated with the aforementioned headers for the project, along with the main function
- makefile
- $create\_infiles.sh$: bash script for the creation of the test sub-directory
- countriesFile: a file with the countries used for the script
- diseasesFile: a file with diseases used for the script
- names - surnames: these files can be used for the script under circumstances

2 Compilation and Running

In order to compile this project, we use a Makefile. Thus, while having an open terminal, type "make" in order to compile the program.Following, in order run the project, type:

./diseaseAggregator -w numWorkers -b bufferSize -i input_dir

In case the user enters a wrong number of arguments, the program prints in the stderr stream the error message and exits with exit code 1. If the arguments given are valid, the program proceeds. All the queries are implemented and their commands can be seen by typing in the command line */help*. All commands begin with the character "/".

3 Data Structures

In this project I have used some of the data structures that were implemented in the previous project

- Simply linked list
- Hash Table
- Red-Black Tree

4 Communication Protocol

4.1 Read sub-directories

The aggregator communicates to the child the number of the directories that it is about to receive. Following that it starts by sending the name of each directory. When the child ends receiving it sends a message that it has finished reading to the parent process, that also means that the child is about to send the statistics that it has collected from all the subdirectories that was assigned to. This procedure is repeated for all the workers when one is created form the father.

4.2 Queries

The aggregator has a getline stacked in an infinite loop and each worker has a blocked fifo pipe waiting to read any message the father will send. Thus, when we enter the query in the command line, the father process sends to all its workers the query. Then, each worker returns the answer it has found to the parent process which prints it on the terminal window. For the queries with multiple amswers, such as topk-AgeRanges or numPatientAdmissions, the answer is given complete to the father (using string concatenation techniques to form it while still in the child).

4.3 Protocol in signal handler SIGUSR1

Every time the SIGUSR1 is given the handler searches for the new file and adds it to the system. Following that it sends a SIGUSR1 to the parent process to open the fifo linking them and sends the stats through it as follows: country - filename - disease stats etc. This procedure is repeated for every new file the handler finds in the subdirectories assigned to the certain worker.

5 Assumptions

The aggregator works with a default bufferSize which can be no less than 120 bytes. In case a lower value is given from the user the aggregator ignores it and starts with the default.In any other case, that is having a $buffersize \geq 120$, the aggregator works properly.

In case the directories are less than the defined from input workers, the aggregator creates the same number of workers as the number of the sub-directory . Thus each worker is assigned just one sub-directory (possibly optimal distribution, depending on the systems resources).

Signals are all handled.