Lab 7 – File I/O and database interfacing

Exercise: File I/O

Implement a program that reads a file 'sensor.data' and prints the result of this file in table format on screen. The file contains an unknown number of sensor measurements. Every sensor measurement written to file has the format:

<sensor id>:<sensor value>@<timestamp>

The types of sensor id, sensor value and timestamp are given in the 'config.h' file, enclosed to this lab. The file is in binary format. Also implement a small program that generates these files starting from the 'sensor simulator' of the previous lab.

Exercise: MySQL interfacing

Implement a program that connects to the MySql server running on “studev.groept.be”. Login using the account “a13_syssoft” with password “a13_syssoft”. (Connection will only work from within the GroupT network, outside you can test on a local MySQL server) Once connected to the database server, the program creates a new table “yourname” in the existing database “a13_syssoft”. This table should have the following columns:

- id: automatically generated unique id
- sensor_id (INT)
- sensor_value (DECIMAL(4,2))
- timestamp (TIMESTAMP)

The C API of MySQL is required in order to implement this program. You can install the MySql client developer package with the following command:

> sudo apt-get install libmysqlclient-dev

Compile your code as follows:

> gcc yourfile.c $(mysql_config –cflags –libs)

Some information sources:

- A tutorial that will guide you all the way through this exercise: http://zetcode.com/tutorials/mysqlcapitutorial/
- The MySql reference manual discusses the full C API: dev.mysql.com/doc/refman/5.0/en/c-api.html

Exercise: File I/O and MySQL interfacing

Adapt the program of the first exercise such that all sensor measurements read from the file 'sensor.data' are now inserted into the database table created in second exercise.

Experiment with some queries to read sensor data from database**.**

THE SOLUTION OF THIS EXERCISE NEEDS TO BE UPLOADED ON TOLEDO BEFORE THE DEADLINE.

Your solution is only accepted if the following criteria are satisfied:

1. *Compilation with '-Wall -Werror' option generates no warnings or errors*
2. *Running 'cppcheck –enaheble=all' on the source code results in no warnings or errors*
3. *The tests below should not fail.*

Before running the tests, do the following:

- *Program to simulate sensor measurements is named 'sensor_simulator.c'*
- *Make sure the binary file containing sensor measurements is named 'sensor.data'*
- *Program to communicate with database is named 'sensor_db.c'*
- *Needless to mention, restrict your implementation to config.h and sensor_db.h*

Test Case 1: run file'sensor_simulator.c

- *check if sensor.data is created and its size is not zero*
- *grep -Evi "[ ]*[0-9]{1,4}[ ]?:[ ]?[\-]?[0-9]{1,2}.[0-9]{2}[ ]*@.*"*

Test Case 2: run file 'sensor_db.c'

- *check if the implementation of all methods is correct and gives result as expected*

DO NOT upload code that doesn't satisfy these criteria.