LAB2-C

OPERATING SYSTEMS-DR. LÉONARD JANER

2021-2022

MARCH 2022

TecnoCampus
Escola Superior
Politècnica

Centre adscrit a la

Universitat
Pompeu Fabra
Barcelona

LAB2-C

Objectives

Test your learnings on threads synchronization and communication on a LINUX environment.

Background / Scenario

In this lab, you will work with threads to share informacion in a synchronized procedure

Required Resources

- DEVASC VM, properly configured

Instructions

Part 1: **THE PROTOCOL**

You have to program a threads' based application named **lab2.c** with the following purpose.

- THREAD1 must be created
- THREAD2 must be created
- THREAD3 must be created
- THREAD1 will read the **quotes.txt** file (from the previous lab) (you have to copy the file again on the current repository)
- THREAD2 will select the number (with a random function) of the message
- THREAD3 will write the message on screen terminal
- MAIN program (thread) will receive as an input argument (using argc/argv) the number of iterations of the process (the number of messages to write on terminal)
- The THREADS must be synchronized using mutex and conditional variables (NO SEMAHORES that will be used on LAB3-C) and share the information for the required communication

Part 2: **GENERAL RECOMMENDATIONS FOR ALL THE PROGRAMS**

Some recommendations:

- Do not use **magic numbers** on your code. Every numerical and/or static value on your code must be defined using the convenient **#define** directive.
- It is a good practice to initialize (always) all the variables, with a default value (event vectors, arrays, …) before you use them. One solution could be to use **memset** when convenient.
- Once your program / process is finished you must freeze the previously allocated resources. If you are using dynamic memory allocation on your program, the **free** system call must be used before exiting your program.

Part 3: **HOW TO ISSUE THE SOLUTION**

You have to provide:

- Your code on the repository.
- On the report you must explain and issue screenshots showing the compilation process to obtain the executable files. If you are using make tool for that process the MAKEFILE must be provided. If the compilation process is not well documented, the program could not be tested.

- Screenshots and explanation of the compiling process, on the report section. The code must be explained on the report. It is not enough to have a code with comments embedded, you have to explain the main structure of the solution, the functions, the parameters. It is as much important the way you write documentation than the way you code.
- Screenshots and explanation of the execution process (more than one example, just as a kind of proof), on the report section.
- Comments on the relevant parts of the code. What you considered must be explained, on the report section.
- On the repository you have to have only one directory. All the files, programs, everything must be on the same directory.