Birzeit University - Faculty of Engineering and Technology
Electrical & Computer Engineering Department - ENCS4330
Real-Time Applications & Embedded Systems - 1^{st} semester - 2024/2025

Project #2
Interprocess communication techniques under Linux
Due: December 1, 2024

Instructor: Dr. Hanna Bullata

File management Simulation

We would like to build a multi-processing application that simulates the creation and handling of csv files (comma-separated files) and that takes advantage of the IPC techniques. The simulation can be explained as follows:

- A user-defined number of file generators will generate the csv files. Each csv file generator will generate a single csv file every random amount of time that belongs to a user-defined range. Assume the default number of file generators is 5 if not provided by the user.

  The generated csv files will have random number of rows and random number of columns and both belong to user-defined ranges. If not specified, the default number of rows is 10,000 and default number of columns is 10. The content of these rows and columns are decimal values that are randomly generated within a user-defined range. Some of the decimal values might not get generated and thus remain missing based on a user-defined miss-percentage provided by the user.

  The format of the file name of the csv files is a `serial_number.csv` where `serial_number` should be replaced by a serial number starting from 0 (e.g. files will be named `0.csv`, `1.csv`, ...). Assume that the initial location for the generated csv files is called the `home directory`. The total number of generated csv files so far must be reported in a shared location.

- A user-defined number of csv file calculators are responsible to calculate the average per column for the previously generated csv files. If for a certain row and column combination a value is missing, it should be excluded from computing the average for that column. Once done, the calculator must record the number of rows and average per column for the csv file in a shared location.

  A csv file calculator will handle one generated csv file at a time. The total number of csv files handled by calculators must be reported in a shared location.

- When a csv file calculator is done with a particular file, a csv file mover is responsible to move that file to directory named `Processed` located under the home directory. If that directory doesn't initially exist, a mover must create it. The number of csv file movers is user-defined but the default number is set to 10 if unspecified by the user.

- 3 types of inspectors are involved in the simulation:

  - Type 1 inspectors: These inspect the home directory and any csv file that is older than a user-defined value and still not handled by a calculator is moved to a directory named `UnProcessed`. If that directory doesn't initially exist, an inspector must create it. The total numbers of unprocessed csv files must be reported in a shared location.
  - Type 2 inspectors: These inspect the `Processed` directory and all csv files that are older than a user-defined value must be moved to directory named `Backup`.

The number of files moved to the `Backup` directory must be reported in a shared location.

- Type 3 inspectors: These inspect the `Backup` directory and all csv files that are older than a user-defined value must be deleted. The number of deleted files must be reported in a shared location.

The number of inspectors for each of the above types is user-defined but default values must be assumed in case they are missing.

- While the simulation is running, the minimum and maximum averages must be reported in a shared location (with the column number and file name).
- The simulation ends if any of the following is true:
  - The number of files that were processed exceeds a user-defined threshold.
  - The number of files that were not processed exceeds a user-defined threshold.
  - The number of files that were moved to the `Backup` directory exceeds a user-defined threshold.
  - The number of files that were deleted exceeds a user-defined threshold.
  - The simulation has been running for more than a user-defined amount of time (in minutes).

What you should do

- Implement the above problem on your Linux machines using a multi-processing approach. Make sure the created processes consume CPU time when needed only.
- Compile and test your program.
- Check that your program is bug-free. Use the `gdb` debugger in case you are having problems during writing the code (and most probably you will :-). In such a case, compile your code using the `-g` option of the `gcc`.
- In order to avoid hard-coding user-defined values in your programs, think of creating a text file that contains all the values that should be user-defined and give the file name as an argument to the main program. That will spare you from having to change your code permanently and re-compile.
- Use graphics elements from opengl library in order to best illustrate the application. Nothing fancy, just simple and elegant elements are enough.
- Be reaslistic in the choices that you make!
- Send the zipped folder that contains your source code and your executable before the deadline. If the deadline is reached and you are still having problems with your code, just send it as is!