Temperature stabilization (IPC)

Due Monday by 11:59pm **Points** 100

GitHub invitation: https://classroom.github.com/a/lLZ57yI5

This is a Linux/Unix OS assignment. It is not an OS/161 Assignment.

Description

In this assignment, you will implement a basic temperature-stabilization system using processes. Their system has multiple processes and they will communicate using inter-process communication methods. You will implement a solution to the problem using **one of the following three** methods for process communication:

1. Message-passing
2. Shared memory
3. Internet stream (TCP) sockets for communication.

You choose the communication method that you want to use.

A temperature-stabilization system consists of 5 processes. Four external processes that communicate their individual temperatures to a central process, which in turn will reply with its own temperature and will indicate whether the entire system has stabilized. Each process will receive its initial temperature upon creation and will recalculate a new temperature according to two formulas:

External processes:

$$externalTemp \leftarrow \frac{(3 \cdot externalTemp + centralTemp)}{5}$$

Central process:

$$centralTemp \leftarrow \frac{2 \cdot centralTemp + \sum_{i=1}^{4} externalTemp_i}{6}$$

Initially, each external process will send its temperature to the central process. If all four temperatures are exactly the same as those sent by the four processes during the last iteration, the system has stabilized. In this case, the central process will notify each external process that it is now finished (along with the central process itself), and each process will output the final stabilized temperature. If the system has not yet become stable, the central process will send its new temperature to each of the outer processes

and await their replies. The processes will continue to run until the temperature has stabilized.

Each external process will be uniquely identified by a command-line parameter. The first parameter to each external process will be its initial temperature, and the second parameter will be its unique number: 1, 2, 3, or 4. The central server will be passed one parameter, i.e., its initial temperature. Name your external processes **external** and the central server **central**.

Implementation details

- Write a Bash script or program that runs you solution.
- The central server can be a multi-connection server. Here, you can choose to implement the multi- connection by using child processes.
- Don't worry about killing Zombies. Let them be.
- The simplest way to start the implementation is to use the sample source code that has been provided as examples in Canvas. Some helpful source-code examples are:
- socket_forkhandlers.zip
- ipc_code.zip

What to submit

Add/Commit/Push the source code of your program into the GitHub repository associated to the assignment. Do not upload executable files or temporary files that result from the compilation process.

Describe the execution of your program in the repository's README.md file. Here, include figures of relevant screenshots showing the execution of the entire system. For example, if you run each process in a separate terminal, you can capture all terminals working side-by-side into a single screenshot. Submissions without a README.md and screenshots showing the program's execution will suffer a 20-point deduction in the grade in the assignment grade.