Advanced and robot programming – Final project

Saporetti Chiara – s4798994

General description

The goal is to simulate a network of multi-process systems, all identical, each one running on a machine in the same LAN, connected through sockets, exchanging a data token at a specified and measured speed.

Everything is implemented on a single machine simulating a distributed implementation. An artificial delay (it will be called waiting time) is used to mimic the network communication delays. The machine contains 4 Posix processes:

- S: Receives console messages as Posix signals.
- P: Receives tokens, computes and sends an updated token after a given delay.
- G: Receives tokens, sends them to P.
- L: Writes log file.

The processes are linked through pipes and a socket

Pipes:

- S-P: Used to send actions to be performed from S to P. The actions will then be sent to be logged.
- G-P: Used to send tokens from G to P. The tokens will then be sent to be logged.
- P-L: Used to send strings from P to L. The strings are for logging purposes.

Socket:

- G-P: Used to send tokens from G to P. P works as the server, G as the client. The tokens will then be sent to be logged.

Files description

Main.c

The main needs some arguments: the log file to which it must write to, the ip address of the computer, the RF, the path to the node G, the number of ports, the original tokem, the waiting time and the path to the two files that contain the PIDs of P, G.

It initializes and creates the pipes. It also asks the user for a command (START, STOP, DUMP-LOG). Than there is the first fork:

- **Process P**: saves its pid to the external file. Then, in a while loop, it creates a select between G and S. In the first loop (since it only needs to do it once at the beginning), it creates a socket to G and a string containing the original token. All the other loops have the same structure: the select may choose either pipe GP or pipe SP.
    If GP: reads the pipe and creates a string containing the received token. Then it creates a socket to G, measures the time to update DT and computes the new token. It finally writes the new token to G through the socket and creates a string containing the new token.
    If SP: reads the pipe and creates a string containing the received action to perform.
    Finally it writes the strings it created on pipe LP.
- Second fork:
    - **Process L**: it opens the log file, reads pipe LP for log information and writes the strings it received to the file. Finally it closes the file.
    - Third fork:
        - **Process G**: saves its pid to the external file. Then it creates the content of the node by making an execcl to the G.c file (that will be described later) with the correct information.
        - **Process S**: reads P and G PIDs from the external files and stops them. Then, in a while loop, it creates the select for pipe SP. It then reads from terminal the action that needs to be performed.
            If command is 1 (START) it sends a SIGCONT signal with the PIDs of G and P, thus starting them.
            If command is 2 (STOP) it sends a SIGSTOP signal with the PIDs of G and P, thus stopping them.
            If command is 3 (DUMP-LOG) it writes the content of the log file on the terminal.

G.c

In a while loop, it connects to the socket to P and reads the token from it. Then creates the select for pipe GP and writes the received token on the pipe.

functions.h

This file provides the functions for the correct functioning of the code. The functions will not be analyzed here since their content is already described in the c codes.

bash_script.sh

This file provides the series of terminal command needed for the correct configuration and running of the code. It should be modified before trying the code.

config_file.cfg

This file provides the values to be passed as arguments to the main function of main.c. It should be modified before trying the code.

Pid_G.txt

This file contains the PID of the process G.

Pid_P.txt

This file contains the PID of the process P.