Entregable Trabajo Práctico N° 2

Tomás Vidal
Sistemas Operativos y Redes
Facultad de Ingeniería, UNLP, La Plata, Argentina.
14 de Octubre, 2024.

I. Problemas presentados

I-A. Problema 10

I-B. Problema 11

Se debe crear un programa que pueda actuar de *dos formas diferentes*, como: **proceso A** o **proceso B**. Ambos deben poder acceder a un *bloque de memoria compartida*, para poder leer y escribir datos en común. Ambos procesos deben mantener una cierta sincronía, ya que uno no puede leer mientras el otro escribe y viceversa.

II. Problema 10

En la figura 1 se presenta el diagrama de flujo que explica la lógica general implementada en el código. Principalmente se hace uso de una pila, para almacenar los PID de los procesos hijos que se van creando. Se registran las señales: **SIGUSR1**, **SIGUSR2** y **SIGTERM**.

II-A. manejador de SIGUSR1

Se encarga de crear un proceso hijo con *fork*, que ejecuta de manera indefinida un buclde que imprime su PID y el de su proceso padre cada un segundo.

II-B. manejador de SIGUSR2

Se encarga de crear un proceso hijo con *fork*, que imprime en pantalla su PID y luego ejecuta el comando: "*ls –color* ."

II-C. manejador de SIGTERM

Se encarga de terminar los procesos hijos a razón de uno por segundo (para esto es que se implementó la pila). Luego termina el proceso padre.

III. Problema 11

III-A. Algorimo implementado

En la figura 2 se muestra el flujo general del programa. El diagrama es *abierto*, en sentido de que hay caminos en paralelo, esto se debe a que el programa tiene un comportamiento concurrente, ya que se registran manejadores de señales que esperan a ser llamados por **SIGUSR1** y **SIGTERM**.

III-B. Funciones principales

- *handle_termination:* Manejador que se encarga de liberar los recursos y terminar ambos procesos.
- *handle_write_read_a:* Realiza la escritura y lectura acorde a las especificaciones del **proceso A**.
- *handle_write_read_b:* Realiza la escritura y lectura acorde a las especificaciones del **proceso B**.
- *cypher:* Función de encriptación de datos.
- *decypher:* Función de desencriptación de datos.
- *setup_shared_memory:* Prepara la memoria compartida para ser usada.
- *save_pid_to_shared_mem:* Guarda en la memoria compartida el PID del proceso actual (A o B).
- *get_pid_from_shared_mem:* Obtiene el PID del proceso opuesto (A o B).
- *destroy_memory_block:* Libera la memoria compartida.

III-C. Explicación general del algorimo

Se registran los manejadores para la señal **SIGUSR1** en ambos procesos; estos manejadores se encargan de la escritura y lectura correspondiente al proceso. Al incio el *proceso A* se envía a sí mismo la señal **SIGUSR1**, de modo que si el *proceso B* existe, comienza un bucle entre ambos procesos de lectura y escritura, ya que al finalizar estas operaciones, cada proceso envía al otro la misma señal **SIGUSR1**.
*Para poder enviar señales entre ambos procesos, se emplea la memoria compartida para almacenar el PID correspondiente a ambos procesos, puesto que se guarda un arreglo de **int** y los PID son del mismo tipo.*
Al comienzo se definen los 20 **int**, que conforman las palabras 'holamundooholaamundo', encriptadas con las especificiones del problema (es decir son *números del 0 al 26*, que representan las letras de la **a** a la **z**), posteriormente en la escritura de ambos procesos, se escriben estos datos nuevamente encriptados con la función provista $f(x)$, y cuando se leen se desencriptan con la función $f^{-1}(x)$

IV. Diagramas

Se adjuntan junto al presente las figuras de los diagramas de flujo, para que el lector pueda analizarlas precisamente.

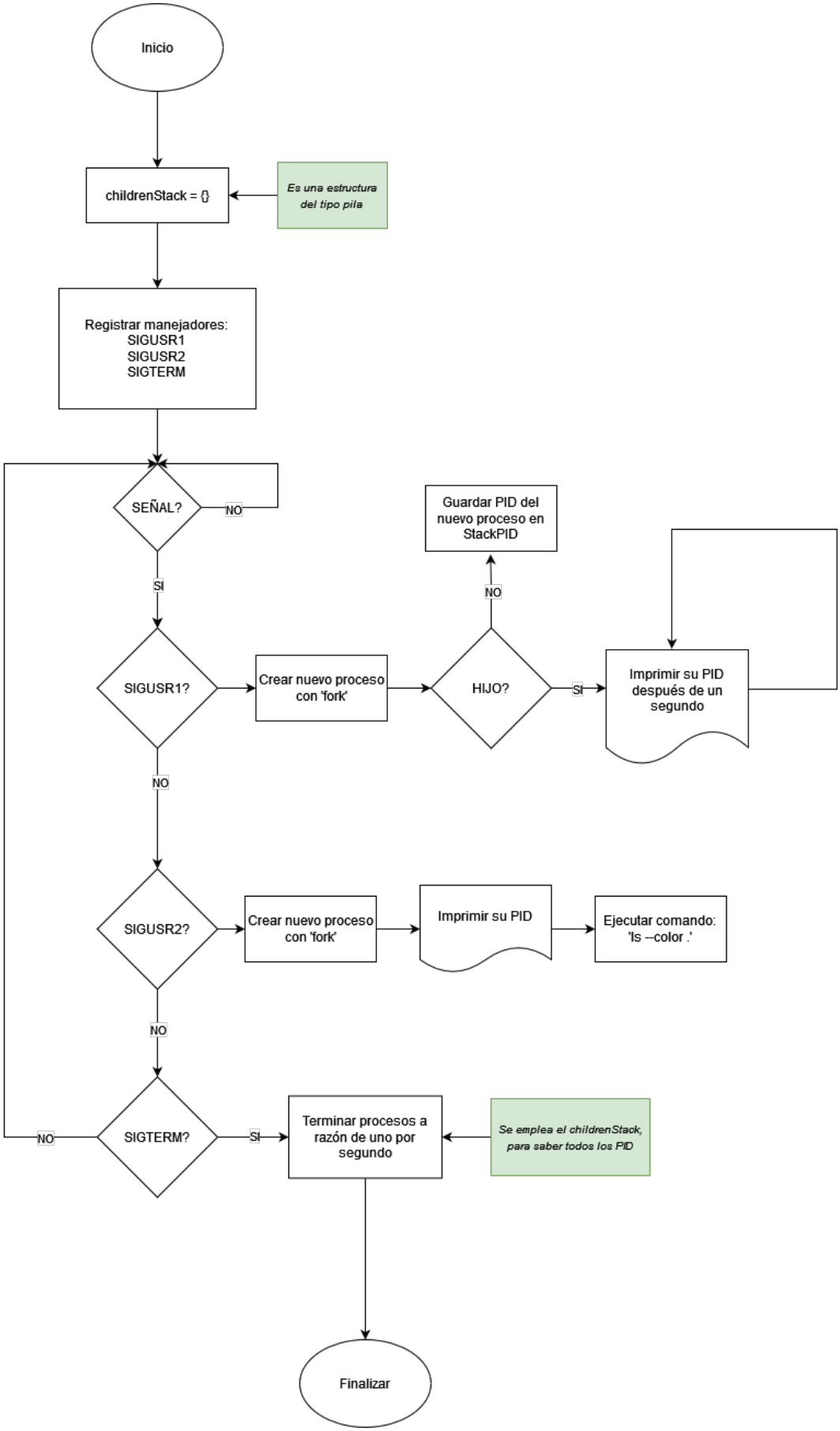

Fig. 1. Diagrama de flujo del problema 10.

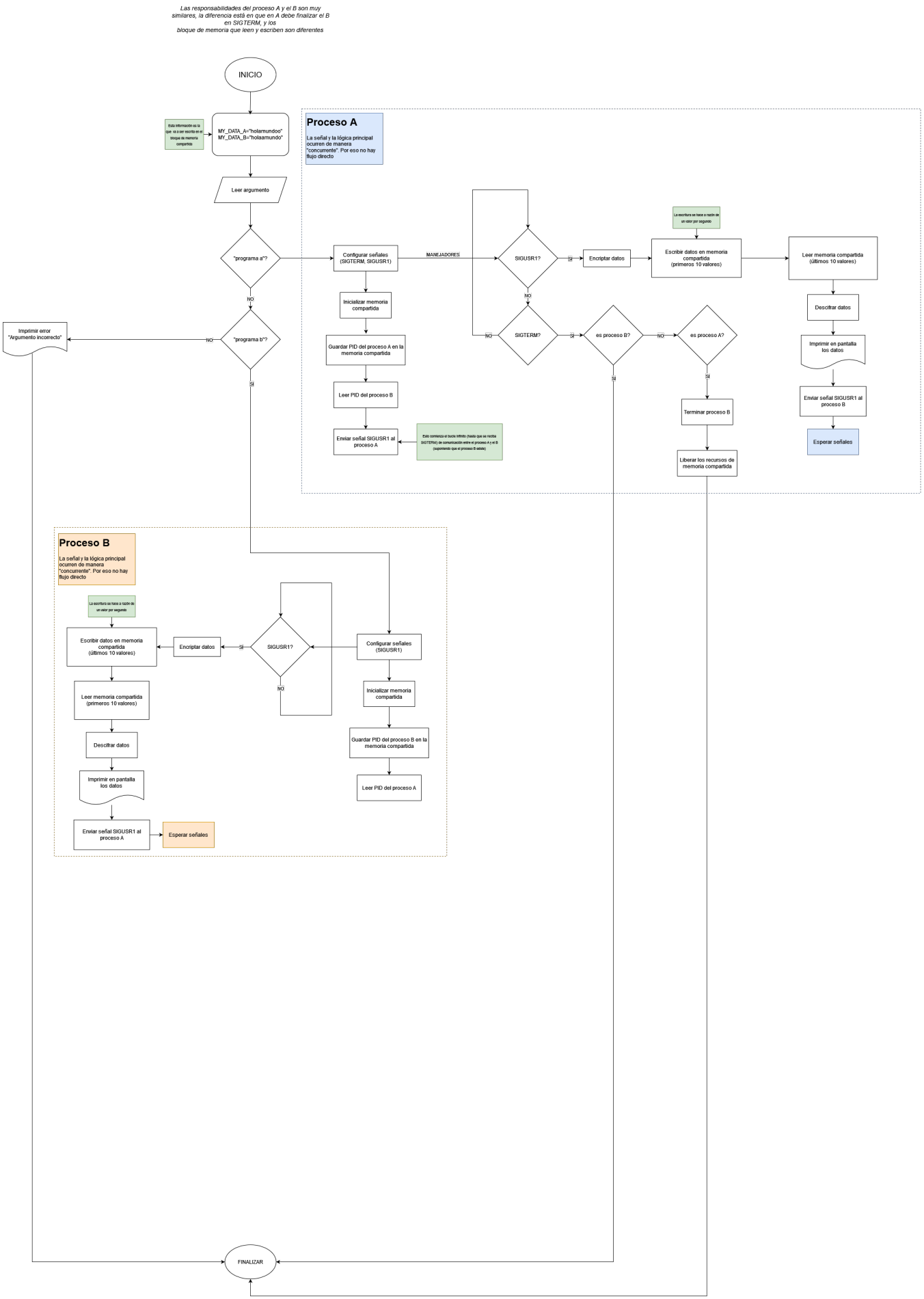

Fig. 2. Diagrama de flujo del problema 11.