INFORME – Taller de Procesos y Comunicación entre Procesos

Autor: Julián Pérez

1. Introducción

En el presente taller se desarrolla una aplicación en lenguaje C que permite la lectura de dos archivos de texto con números enteros, con el fin de calcular la suma de los elementos en cada uno y una suma total. El programa está diseñado utilizando los conceptos de creación de procesos (fork) y comunicación entre procesos mediante tuberías (pipe). De este modo, se refuerza la comprensión del modelo de procesos en sistemas operativos tipo Unix y la coordinación entre ellos para la ejecución de tareas concurrentes.

2. Objetivo

Aplicar los conceptos de creación y sincronización de procesos, así como la comunicación entre ellos, mediante el uso de llamadas al sistema como fork(), wait() y pipe().

3. Desarrollo

El programa implementado recibe por línea de comandos dos archivos de texto y sus respectivas cantidades de elementos (N1 y N2). Cada archivo contiene una lista de números enteros.

1. Se utilizan dos llamadas a malloc() para reservar espacio dinámico en memoria para los arreglos A y B.
2. Los datos se cargan desde los archivos mediante una función leer_arreglo().
3. Se crea un pipe() para que los procesos puedan enviarse información (sumas parciales y total).
4. A partir de ahí:
   - Se crea un primer proceso hijo (pid1) mediante fork().
   - Este proceso a su vez crea un proceso nieto, encargado de sumar los datos del primer archivo (sumaA) y escribir el resultado en el pipe.
   - Una vez el nieto termina, el proceso hijo crea un segundo hijo, que calcula la suma del segundo archivo (sumaB) y también la envía por el pipe.
   - Finalmente, el proceso hijo calcula la suma total (sumaTotal) y la escribe también en el pipe.

5. El proceso padre espera que los hijos finalicen (wait()), cierra el extremo de escritura del pipe y lee los tres resultados: sumaA, sumaB y sumaTotal.
6. Por último, se imprimen las tres sumas y se libera la memoria reservada.

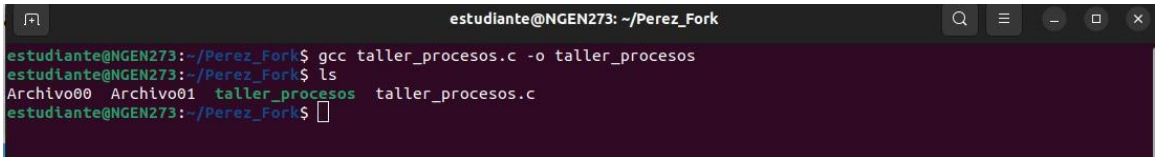
4. Resultados

El programa fue probado con archivos .txt que contienen listas de números enteros.

archivo00.txt
10 20 30 40 50

archivo01.txt
5 15 25 35

Primero creamos el ejecutable del código principal y lo llamamos taller_procesos



Al ejecutar el programa con los argumentos:

La salida fue:

Esto demuestra que cada proceso realizó correctamente su cálculo y que la comunicación por pipe fue exitosa.

5. Conclusiones

Este ejercicio permitió comprender de manera práctica cómo se crean procesos en C y cómo estos pueden comunicarse utilizando tuberías. También se evidenció la importancia de la sincronización mediante wait() para evitar condiciones de carrera. Además, se reforzó el manejo de memoria dinámica con malloc() y el uso correcto de archivos en C. En conclusión, el taller cumplió con su objetivo de aplicar y entender los fundamentos de procesos y comunicación en sistemas operativos.