# Diseño funcional Ejercicio 1

## 1. Abstract

El sistema simula procesamiento en paralelo mediante un proceso padre que crea N hijos, los cuales modifican datos compartidos utilizando memoria compartida y semáforos para sincronización. Cada hijo realiza una tarea distinta hasta que el padre ordena la finalización, mostrando luego el resultado final del procesamiento.

## 2. Lenguaje de implementación

* **Lenguaje elegido:** C
* **Sistema operativo requerido:** Linux (por uso de fork(), shmget(), semget(), etc.)
* **Compilador:** GCC (GNU Compiler Collection)

## 3. Enunciado con el diseño funcional de la problemática

**Objetivo:**

Diseñar un programa que simule procesamiento de datos en paralelo mediante múltiples procesos. La información debe ser compartida entre los procesos hijos, que actuarán simultáneamente sobre los datos usando semáforos para evitar condiciones de carrera.

**Lógica funcional del programa:**

1. **Inicialización**
   * El proceso padre crea un segmento de memoria compartida donde se almacena un vector de datos.
   * Se crea un semáforo para controlar el acceso a la memoria compartida.
2. **Creación de procesos**
   * El padre crea **N procesos hijos** (en este caso, 3).
   * Cada hijo recibe un identificador (ID) y ejecuta una operación específica sobre el vector compartido:
     + Hijo 0: Suma 1 a cada elemento.
     + Hijo 1: Resta 1 a cada elemento.
     + Hijo 2: Multiplica cada elemento por 2.
3. **Sincronización**
   * Para evitar conflictos, cada hijo accede a la memoria compartida **sólo cuando obtiene el semáforo**.
   * Tras completar su tarea, libera el semáforo y duerme quince segundos para simular procesamiento.
4. **Finalización**
   * El padre espera a que el usuario presione ENTER.
   * Cambia un flag (finalizar = 1) en la memoria compartida para que los hijos terminen.
   * Espera la finalización de todos los hijos (wait()).
   * Muestra el resultado final del vector procesado.
5. **Limpieza**
   * Se eliminan la memoria compartida y el semáforo para evitar recursos “basura” en el sistema.

## 4. Diagrama

[Inicio - Padre]

|

v

[Crea memoria compartida]

|

v

[Crea semáforo]

|

v

[Inicializa vector]

|

v

[Crea hijos con fork()]

|

v

<<Cada hijo>>

|

v

[Toma semáforo]

|

v

[Opera sobre vector]

|

v

[Libera semáforo y repite]

<<hasta que finalizar == 1>>

<<Padre>>

|

v

[Espera ENTER del usuario]

|

v

[Setea finalizar = 1]

|

v

[Espera hijos con wait()]

|

v

[Imprime vector final]

|

v

[Limpia semáforo y memoria]

|

v

[Fin]