# Tarea Asincrónica: IPC y Sockets

**Niurka Vanesa Yupanqui**

10 de julio de 2025

---

Durante esta tarea asincrónica sobre IPC y sockets, trabajé con cuatro programas clave que me ayudaron a entender cómo se comunican los procesos y cómo funciona la concurrencia en los sistemas operativos.

Primero, probé el archivo `fork.c`, que crea un proceso hijo usando la función `fork()`. Al ejecutarlo, noté que tanto el proceso padre como el hijo corrían el mismo código, pero con diferentes identificadores. Esto me ayudó a ver cómo el sistema puede hacer que dos procesos trabajen al mismo tiempo de forma independiente.

Después usé `sumPar.c`, que realiza una suma de forma paralela usando varios procesos. Comparado con una suma secuencial, este método fue más rápido, lo cual muestra que dividir el trabajo entre procesos puede hacer que los programas funcionen mejor cuando se trata de tareas repetitivas o pesadas.

Luego ejecuté `multiSharVar.c`, donde dos hilos aumentan una misma variable entera sin sincronización. Al hacerlo varias veces, el resultado cambiaba, lo cual me hizo darme cuenta de que los hilos pueden pisarse entre sí si no se controla el acceso a los datos compartidos. Este experimento me mostró claramente qué es una condición de carrera y por qué es importante evitarla.

Finalmente, trabajé con `multiEcho.c`, donde dos hilos copian el contenido de un archivo a otro utilizando una misma variable tipo `char`. Aunque parecía funcionar, a veces salían errores o comportamientos raros. Esto me hizo pensar en cómo los hilos, si no están bien organizados, pueden generar errores similares a los que ocurren con sockets si no se manejan bien.

Esta práctica me ayudó a comprender mejor cómo se relacionan los procesos, los hilos y la comunicación dentro de un sistema. Al ejecutar y analizar cada programa, reforcé lo aprendido en clase y pude observar con más claridad cómo funcionan internamente estos mecanismos en un sistema operativo real.