Nombres de Integrantes: Washington Parra, Giovanni Sambonino, Diego Contreras.
Grupo #12

pa05-Diseño e implementación una cadena de procesamiento con mecanismos de IPC

Introducción

La finalidad de este proyecto es diseñar e implementar una cadena de procesamiento de imágenes, utilizando técnicas de comunicación entre procesos (IPC) y mecanismos de sincronización. El sistema conforma cuatro programas vitales que usan memoria compartida como va de comunicación. El publicador, encargado de cargar imágenes BMP desde el sistema de archivos, el desenfocador, que aplica un filtro de desenfoque a la parte superior de la imagen, el realzador que realza los bordes en la parte inferior y el combinador, programa que se encarga de combinar las dos secciones procesadas para generar la imagen final.

Problemática

La problemática tiene como desafío diseñar un sistema que procese imágenes en paralelo usando IPC. Se trata de leer una imagen BMP publicada por un publicador a una memoria compartida, dividirla en dos partes y aplicar transformaciones diferentes: desenfoque en la parte superior y realzado de bordes en la parte inferior. Los procesos deben comunicarse mediante memoria compartida y sincronizar el acceso para evitar conflictos. El objetivo es combinar ambas partes procesadas en una sola imagen. La motivación es crear un pipeline de procesamiento eficiente y modular, mejorando el rendimiento mediante concurrencia y memoria compartida.

Diseño de la solución

Publicador.c: El publicador carga una imagen BMP en una memoria compartida creada. Pide al usuario la ruta de la imagen, lee el encabezado y los píxeles con la función readBMP. Usa shm_open para crear un segmento de memoria compartida y mmap para mapearlo. Luego, copia el encabezado y los píxeles de la imagen en la memoria. El proceso se repite en un bucle, permitiendo cargar varias imágenes si el usuario lo desea.

Realzador.c: Aquí se accede a la imagen guardada en memoria compartida por el publicador.c, se mapea la memoria compartida para después leer los datos de la imagen y se usa un filtro de detección de bordes a la mitad inferior de la imagen usando la función applyEdgeDetection(imageOut, edgeFilter) que se aloja en filter.c. Luego, la imagen realzada se almacena en una nueva memoria compartida.

Ex5.c (Desenfocador): Similar al Realzador, se accede a la imagen guardada en memoria compartida por el publicador.c, se mapea la memoria compartida para después leer los datos de la imagen y se usa un filtro de desenfoque a la mitad superior de la imagen usando la función applyDesenfoque que se aloja en bmp.c. Luego, la imagen desenfocada se almacena en una nueva memoria compartida.

Combine.c: Una vez teniendo los dos espacios de memoria compartida creada por el desenfocador y el realzador, el archivo combine.c combina la mitad superior desenfocada y la mitad inferior realzada para formar una imagen completa. Accediendo en ambos espacios de memoria compartida que contienen las imágenes desenfocada y realzada, copia los píxeles de ambas mitades en una nueva estructura de imagen y guarda esta imagen combinada en un archivo local, calculando y asegurando que el tamaño del archivo y el padding estén correctos. Al final se libera la memoria utilizada para almacenar los píxeles.

Limitaciones y como lo resolví

El proyecto en general estuvo complicado en varios aspectos, las limitantes que vi fueron varias.

- Uno de los principales problemas fue que algunos programas no podían mapear correctamente la memoria compartida, generando errores, por lo que tuvimos que agregar verificaciones y mensajes para encontrar los fallos en el programa.
- Otra limitación que tuvimos al realizar fue manejar la memoria compartida, ya que tuvimos que pensar en darle la suficiente memoria a las imágenes. Esto se resolvió usando el tamaño de la imagen como el tamaño de los segmentos de la memoria compartida.
- La limitación mas ligera fue el desconocimiento de uso de funciones como shm_open y mmap ya que no teníamos conocimiento previo sobre cómo gestionar la memoria compartida, manejar permisos y sincronizar el acceso entre procesos, pero esto se soluciono gracias a la capacitación con documentaciones detalladas, ejemplos de uso sobre cómo funcionan estas funciones y videos tutoriales.

Pruebas y Salidas de Pantalla

Para asegurar el correcto funcionamiento del programa, se realizaron diversas pruebas. A continuación, se presentan las pruebas y los resultados obtenidos:

Publicador

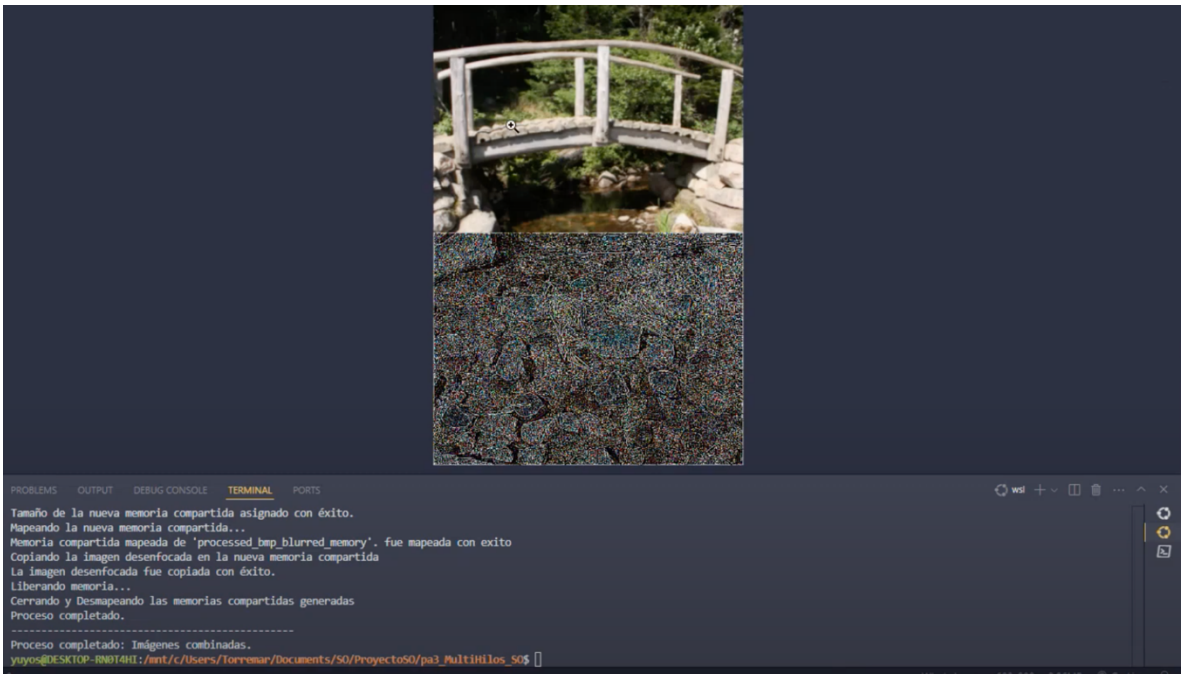
En la siguiente imagen se ve como se usa al archivo publicador.c para poder almacenar la imagen en una memoria compartida solicitando la ruta de la imagen.

Se guardará la imagen en la memoria compartida, permitiendo guardar también otras imágenes.

Filtros(Desenfocador y Realzador) y combinación de la imagen

Se corre un bash en los cuales se usan los programas ex5 y realzador, se usa el wait para esperar ambos programas y despues usar el programa combine para combinar ambas imágenes. Cabe recalcar que para usar el bash hay que dar permisos.

Se mostrará una serie de mensajes que describirán el proceso del desenfocador y realzador y si esta funcionando, para al final obtener una imagen en la carpeta output con desenfoque en la parte superior y realzador en la parte inferior.



Link repositorio: https://github.com/GiovanniSambonino/ProyectoSO
Link video: https://youtu.be/oVnhuCnwPz8