Reporte de la Solución - Programa de Lectura de Archivos con Manejo de Errores en C

1. Problemática a Resolver

El objetivo de este proyecto era implementar un programa en C que simule el comportamiento del comando `cat` de Linux. Este programa debe cumplir dos funciones principales:

- **Leer archivos**: Mostrar el contenido de un archivo especificado como argumento.
- **Leer entrada estándar**: Si no se proporciona un archivo, el programa debe leer desde la entrada estándar.
  Además, se requería que el programa manejara **errores de forma adecuada**, utilizando las funciones POSIX `open()`, `read()`, y `write()`. En caso de error (por ejemplo, si el archivo no existe o no puede abrirse), el programa debía proporcionar mensajes de error claros en la salida estándar de error (**stderr**).

2. Limitaciones y cómo se resolvieron

Durante el desarrollo de este programa, se enfrentó con las siguientes limitaciones:

- **Manejo de errores detallado**:
  - **Problema**: Las funciones POSIX como `open()`, `read()` y `write()` retornan valores negativos en caso de error, pero no proporcionan detalles adicionales directamente sobre el problema ocurrido.
  - **Solución**: Se utilizó la función `strerror(errno)` para convertir el valor de `errno` en una descripción detallada del error, mostrando mensajes claros de error, como por ejemplo: "Error en apertura del archivo: No such file or directory".
- **Manejo de múltiples casos de uso**:
  - **Problema**: El programa debe manejar tanto la entrada estándar como la lectura de archivos, y además asegurar que los errores sean tratados en ambos casos.
  - **Solución**: Se implementó un manejo condicional para detectar cuándo se debe leer desde el archivo (cuando se proporciona un argumento) y cuándo desde la entrada estándar (sin argumentos). Además, se incluyó manejo de errores en ambos escenarios.
- **Compatibilidad con diferentes tamaños de archivos**:
  - **Problema**: Los archivos grandes podrían sobrecargar la memoria si no se gestionan correctamente los buffers.
  - **Solución**: Se definió un tamaño de buffer (`BUFFER_SIZE`) razonable para leer los datos en bloques de hasta 1024 bytes por iteración, evitando así problemas de

memoria con archivos grandes.

3. Salidas de Pantalla de las Pruebas Realizadas