Instituto Tecnológico de Costa Rica

Campus Tecnológico Central Cartago

Escuela de Ingeniería en Computación

Principios de Sistemas Operativos

Profesor: Kenneth Obando Rodriguez

Tarea #4: Sistema de Archivos

Jose Miguel González Barrantes || 202308756

Erick Abarca Calderón || 2022296303

II Semestre 2025

Índice:

Introducción:

Esta tarea consiste en la implementación de un sistema de archivos simple en lenguaje C que viene a simular las operaciones básicas de gestión de archivos que son parte fundamental de un sistema operativo. Este consiste en simular una estructura de almacenamiento basada en bloques de memoria de tamaño fijo, en donde se permite también funciones importantísimas como sería la creación, escritura, lectura y eliminación de archivos. Desarrollado como parte del curso Principios de Sistemas Operativos del Tecnológico de Costa Rica, este trabajo demuestra los conceptos fundamentales de gestión de almacenamiento y organización de datos. La implementación incluye un directorio raíz completamente virtual donde se almacenan los datos de los archivos y una tabla de asignación de bloques.

El sistema funciona a partir del uso de una serie de comandos que como se mencionó anteriormente, entre algunas de las funcionalidades implementadas se encuentran la creación de archivos con tamaño específico, escritura de datos en posiciones determinadas, lectura de contenido y eliminación de archivos con liberación automática de recursos. Adicionalmente el diseño también incluye mecanismos de manejo de errores para situaciones como podrían ser: archivos inexistentes, espacio insuficiente o operaciones fuera de límites. Estas tres últimas funcionalidades no son exactamente parte de la versión inicial de la solución pero se añadieron pues se consideran de gran importancia para el debugging, trazabilidad y mantenibilidad.

Arquitectura

El sistema sigue una arquitectura modular bien organizada en distintas capas. En la parte superficial se puede encontrar la capa de interfaz que maneja todas las distintas interacciones con el usuario, seguida por el parser que es el responsable en procesar los comandos de entrada que el usuario digite. Después la capa de file manager actúa como intermediario entre la interfaz y la lógica de negocio, mientras que la capa de “operations” contiene toda la funcionalidad principal del sistema. Y por último, la capa de datos gestiona las estructuras de almacenamiento en memoria.

Las estructuras de datos principales incluyen FileSystem como contenedor global del estado del sistema, FileEntry para almacenar los metadatos de cada archivo, y Block como unidad básica de almacenamiento de 512 bytes. FileEntry tiene como finalidad el mantener información como por ejemplo el nombre, tamaño, cantidad de bloques asignados y los índices específicos de estos bloques. Que son atributos extremadamente importantes para poder realizar operaciones, búsquedas y análisis a partir de estos datos y cómo estos van cambiando durante la ejecución.

La decisión de implementar una arquitectura en capas surgió de la necesidad de separar claramente las responsabilidades, por lo tanto la lógica y la gestión de almacenamiento están separadas entre sí. Esta separación permite que cada módulo se enfoque en una tarea específica: el parser solo se preocupa por interpretar texto, las operaciones contienen pura lógica de archivos, y el file manager actúa como intermediario y es el manejo en sí de los archivos. Esta es de las mejores maneras de ordenar esta tarea, pues cada segmento o parte del sistema está separado entre sí, en dónde está la del manejo del sistema de archivos, parser, operaciones, etc. Esto nos permite entender de una mejor manera cómo se relacionan y trabajan entre sí. Además nos permite mejorar aspectos como la mantenibilidad, escalabilidad, estabilidad y debugging durante el desarrollo. A continuación se presenta una foto de la arquitectura y el distinto ordenamiento de los archivos.

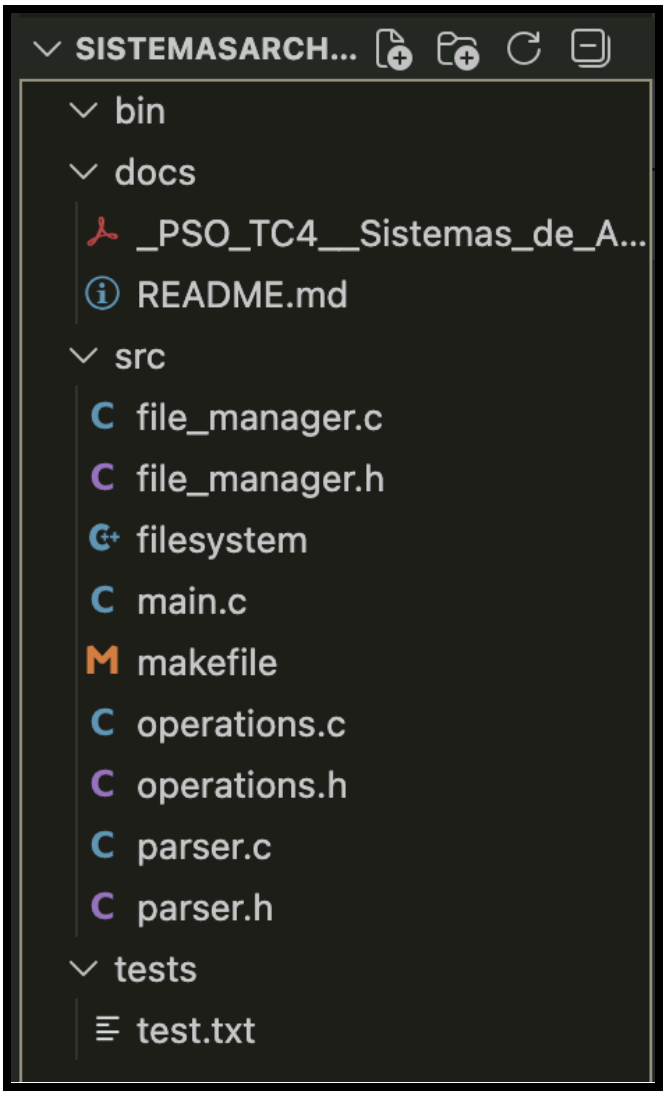

Figura #1 Ordenamiento de los distintos archivos

Como se puede observar en la imagen, se puede ver la carpeta tests que es la que incluye las distintas pruebas para poder demostrar el correcto funcionamiento del programa. La carpeta src es aquella que contiene gran parte de la tarea, pues contiene los módulos principales del programa. Por último la carpeta docs que es la que tiene la documentación importante del proyecto, desde el PDF con la presentación de la tarea a realizar, además del respectivo README.md y por último el archivo de documentación

Estructuras:

1. **FileSystem**

FileSystem es la estructura principal que actúa como el núcleo del sistema de archivos completo. Contiene un arreglo de FileEntry para gestionar hasta 100 archivos simultáneamente y un arreglo de Block para representar los 2048 bloques de almacenamiento disponibles. Mantiene un registro del espacio total utilizado y el conteo actual de archivos activos en el sistema. Esta estructura centralizada permite una gestión unificada de todos los recursos del sistema.

2. **FileEntry**

FileEntry funciona como la tabla de metadatos para cada archivo individual dentro del sistema. Almacena el nombre del archivo, su tamaño en bytes y la cantidad de bloques que ha asignado. Incluye un arreglo que guarda los índices específicos de los bloques de memoria asignados a ese archivo en particular. El campo is_used indica si la entrada está actualmente ocupada o disponible para reuse.

3. **Block**

Block representa la unidad fundamental de almacenamiento en el sistema con un tamaño fijo de 512 bytes. Cada bloque contiene un arreglo de caracteres que almacena los datos reales del archivo en esa posición. La bandera is_used indica si el bloque está actualmente asignado a algún archivo o se encuentra disponible. Esta estructura básica permite la gestión eficiente del espacio de almacenamiento simulado.

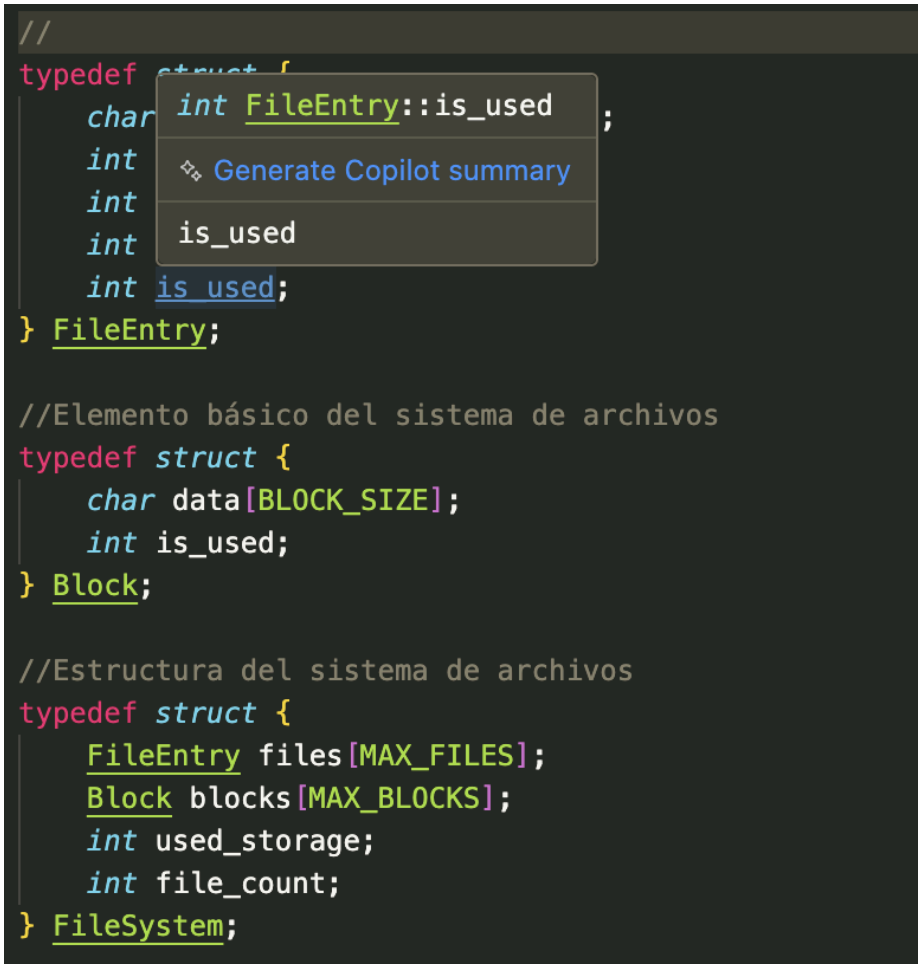

Figura #2 Estructuras utilizadas en la tarea #4