Universidad Metropolitana

Departamento de Gestión de Proyectos y Sistemas

Sistemas Operativos

Trimestre: 2425-2

Preparadores: Sofía León y Andrés Martin

Informe Proyecto 2: Simulador Virtual de Sistema de Archivos con Gestión de Permisos y Asignación de Bloques

Alejandra Carrera

Gianfranco Mongiello

Caracas, 20 de Marzo de 2025

Objetivo del Proyecto

El objetivo fue desarrollar un simulador de sistema de archivos que permitiera comprender y aplicar conceptos fundamentales como la gestión jerárquica de directorios, la asignación de bloques de almacenamiento, y la persistencia de datos. El simulador debía ofrecer una interfaz gráfica dinámica para visualizar la estructura de archivos, el estado del disco virtual (SD), y una tabla de asignación de bloques, además de operar en dos modos: Administrador (CRUD completo) y Usuario (solo lectura).

Funcionalidades Clave

1. Visualización de la estructura de archivos:
    - Uso de JTree para representar jerarquías de directorios y archivos.
    - Actualización en tiempo real al crear, modificar o eliminar elementos.
2. Simulación del disco virtual (SD):
    - Representación visual de bloques ocupados (rojos) y libres (verdes).
    - Asignación encadenada de bloques para archivos, evitando fragmentación externa.
3. Operaciones CRUD:
    - Crear, renombrar y eliminar archivos/directorios (solo en modo Administrador).
    - Validación de espacio disponible en el SD antes de crear archivos.
4. Persistencia de datos:
    - Guardado y carga del estado del sistema en formato JSON.
5. Modos de usuario:
    - Restricción de operaciones en modo Usuario para evitar modificaciones no autorizadas.

Clases y Métodos Principales

1. Clase SistemaArchivo

Propósito: Gestiona la estructura del sistema de archivos, incluyendo directorios raíz, operaciones CRUD, y persistencia.

Atributos Clave:

- Directorio raiz: Directorio raíz del sistema.
- MemoryManager memoryManager: Administra la asignación y liberación de bloques en el SD.

Métodos Clave:

- crearDirectorio(String nombrePadre, String nombreDirectorio): Crea un directorio dentro de otro existente.
- eliminarDirectorio(String nombrePadre, String nombreDirectorio): Elimina un directorio y libera sus bloques.
- crearArchivo(String nombreDirectorio, String nombreArchivo, int tamano): Asigna bloques a un archivo nuevo.
- guardarEstado(String rutaArchivo): Serializa el sistema en JSON.
- cargarEstado(String rutaArchivo): Reconstruye el sistema desde JSON, corrigiendo referencias de bloques.

2. Clase Directorio

Propósito: Representa un directorio en el sistema, almacenando subdirectorios y archivos.

Atributos Clave:

- Lista<Directorio> subdirectorios: Lista de subdirectorios.

- Lista<Archivo> archivos: Lista de archivos contenidos.

Métodos Clave:

- agregarSubdirectorio(Directorio subdirectorio): Añade un subdirectorio.
- eliminarSubdirectorio(String nombre, MemoryManager mm): Elimina un subdirectorio y libera recursos recursivamente.
- buscarSubdirectorio(String nombre): Verifica la existencia de un subdirectorio.

3. Clase Archivo

Propósito: Representa un archivo con sus metadatos y bloques asignados.

Atributos Clave:

- String nombre: Nombre del archivo.
- Lista<Bloque> bloquesAsignados: Bloques ocupados en el SD.
- int tamañoBloques: Cantidad de bloques asignados.

Métodos Clave:

- setBloquesAsignados(Lista<Bloque> bloques): Asigna bloques al archivo.
- getBloquesAsignados(): Retorna la lista de bloques ocupados.

4. Clase MemoryManager

Propósito: Administra la asignación y liberación de bloques en el SD.

Atributos Clave:

- Lista<Bloque> todosLosBloques: Todos los bloques del SD.
- Queue<Bloque> bloquesLibres: Cola de bloques disponibles.

Métodos Clave:

- asignarBloquesEncadenados(int cantidad): Asigna bloques contiguos a un archivo.
- liberarBloques(Lista<Bloque> bloques): Libera bloques y los reencola como libres.
- reconstruirColaBloquesLibres(): Reconstruye la cola tras cargar desde JSON.

5. Clase Simulador (Interfaz Gráfica)

Propósito: Proporciona una interfaz gráfica para interactuar con el sistema de archivos.

Componentes Clave:

- JTree: Muestra la jerarquía de directorios/archivos.
- JPanel panelsd: Representa visualmente los bloques del SD.
- JTable jTable1: Tabla de asignación de archivos (nombre, bloques, primer bloque, cadena).

Métodos Clave:

- actualizarJTree(): Actualiza el árbol tras operaciones CRUD.
- configurarPanelSD(): Pinta los bloques ocupados/libres en el SD.
- actualizarTablaAsignacion(): Actualiza la tabla con datos de archivos.

Resolución de Problemas Clave

1. Eliminación incorrecta de archivos:
    - **Causa:** La clase Lista no manejaba correctamente la eliminación del primer elemento.
    - **Solución:** Se corrigió el método deleteIndex para actualizar la referencia head al eliminar el primer nodo.

2. Actualización del SD al cargar desde JSON:
    - **Causa:** Los bloques de serializados no coincidían con las instancias del MemoryManager.
    - **Solución:** Se implementó reemplazarBloquesEnArchivos() para vincular bloques cargados con los del sistema actual.

6.Clase LoggerSistema

Propósito: La clase LoggerSistema se encarga de registrar acciones del sistema en un archivo de log (log.txt), incluyendo la información del usuario y la acción realizada, junto con una marca de tiempo.

Atributos clave:

- LOG_FILE: Una constante String que almacena el nombre del archivo donde se guardarán los registros de acciones.

Métodos clave:

- registrarAccion(String usuario, String accion):
    - Recibe el nombre del usuario y la acción realizada.
    - Obtiene la fecha y hora actual en el formato yyyy-MM-dd HH:mm:ss.
    - Escribe la información en el archivo log.txt, agregando una nueva línea con cada registro.
    - Maneja posibles excepciones de entrada/salida (IOException).

Conclusión

El simulador cumple con los requerimientos funcionales, permitiendo gestionar un sistema de archivos virtual con persistencia en JSON, asignación encadenada de bloques, y una interfaz gráfica intuitiva. Las correcciones en la estructura Lista y la reconstrucción de referencias aseguran integridad en operaciones CRUD y carga de datos. Este proyecto sirve como herramienta educativa para entender la gestión de almacenamiento en sistemas operativos

Clases y Métodos Auxiliares

1. Clase Nodo

Responsabilidad: Representar la unidad básica de almacenamiento en estructuras como listas enlazadas y colas.

Atributos Principales

Atributo	Tipo	Descripción
data	T	Dato almacenado en el nodo.
next	Nodo	Referencia al siguiente nodo en la lista.

Métodos Clave

Método	Descripción
Nodo(T data)	Constructor. Inicializa el nodo con un dato y next = null.
getData()	Retorna el valor almacenado en el nodo.
setData(T data)	Establece el valor del nodo.
getNext()	Retorna la referencia al siguiente nodo.
setNext(Nodo next)	Establece la referencia al siguiente nodo.

Conclusión

El Nodo es la base para construir estructuras dinámicas como listas y colas, permitiendo almacenar y enlazar datos de manera eficiente.

2. Clase Lista<T>

Responsabilidad: Gestionar una colección de nodos enlazados, facilitando operaciones CRUD (Crear, Leer, Actualizar, Eliminar).

Atributos Principales

Atributo	Tipo	Descripción
head	Nodo	Primer nodo de la lista.
length	int	Número de elementos en la lista.

Métodos Clave

Método	Descripción
insertFirst(T data)	Inserta un nodo al inicio de la lista.
insertLast(T data)	Inserta un nodo al final de la lista.
insertIndex(T data, int position)	Inserta un nodo en una posición específica.
deleteFirst()	Elimina el primer nodo de la lista.
deleteLast()	Elimina el último nodo de la lista.
deleteIndex(int position)	Elimina un nodo en una posición específica.
get(int index)	Retorna el dato del nodo en la posición indicada.

contains(T elemento)	Verifica si un elemento existe en la lista.
esVacio()	Retorna true si la lista está vacía.

Conclusión

La Lista es crítica para almacenar subdirectorios y archivos dentro de un directorio. Su implementación con enlaces simples garantiza flexibilidad y eficiencia en operaciones de inserción/eliminación.

3. Clase Queue<T>

Responsabilidad: Implementar una cola FIFO (First-In, First-Out) para gestionar procesos en orden de llegada.

Atributos Principales

Atributo	Tipo	Descripción
first	Nodo	Primer nodo de la cola.
last	Nodo	Último nodo de la cola.
length	int	Número de elementos en la cola.

Métodos Clave

Método	Descripción
enqueue(T data)	Agrega un elemento al final de la cola.

dequeue()	Elimina y retorna el primer elemento de la cola.
peek()	Retorna el primer elemento sin eliminarlo.
isEmpty()	Retorna true si la cola está vacía.
clear()	Vacía la cola, reiniciando first, last y length.

Conclusión

La Queue es esencial en el MemoryManager para gestionar bloques libres en el SD. Su naturaleza FIFO garantiza una asignación justa y ordenada.

Integración en el Sistema de Archivos

Estructura	Uso en el Proyecto
Nodo	Base para construir Lista y Queue.
Lista	Almacena subdirectorios y archivos en un Directorio.
Queue	Gestiona la cola de bloques libres en MemoryManager.