UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
FACULTAD DE INGENIERÍA

SISTEMAS OPERATIVOS

PROYECTO:
(MICRO) SISTEMA DE ARCHIVOS MULTIHILOS

GRUPO: 6
ALUMNOS:

- **JIMÉNEZ GUTIÉRREZ AXEL URIEL**
- **LÓPEZ REYES ALAM**

PROFESOR: ING. GUNNAR EYAL WOLF ISZAEVICH

FECHA DE ENTREGA: 19 DE MAYO DE 2024

ESTRATEGIA UTILIZADA

El siguiente programa implementa el sistema de archivos 'FiUnamFS', el cual se diseñó para proporcionar una estructura mínima y básica de la gestión de archivos dentro de una imagen de disco. La estrategia seleccionada es implementar un sistema de archivos con un superbloque, un directorio de archivos con un tamaño fijo y clusters para el almacenamiento de datos. Las operaciones principales incluyen la inicialización y posterior validación del sistema de archivos, listar el contenido, hacer copias de archivos desde y hacia el sistema de archivos, y eliminación de archivos.

REQUISITOS DE USO

- Sistema Operativo: Windows 10 y 11
- Lenguaje de programación utilizado: Python 3.

EXPLICACIÓN DE FUNCIONAMIENTO

INICIALIZACIÓN Y VALIDACIÓN

Antes de iniciar con las características implementadas en este sistema de archivos, primero es necesario inicializar el sistema y validar que el archivo de imagen cumple con el formato esperado.

```
if __name__ == "__main__":
    fs = FiUnamFS(r'C:\Users\AlamLR\Desktop\proyectoSO\fiunamfs.img')
    fs.validate_fs()
    main_menu(fs)
```

INTERFAZ DEL MENÚ

Se agregó al programa un menú interactivo que permite al usuario gestionar archivos dentro del sistema FiUnamFS a través de varias opciones. Este menú utiliza hilos para realizar operaciones como listar contenidos, copiar archivos desde y hacia el sistema FiUnamFS, y eliminar archivos, todo sin bloquear la interacción principal del usuario con el menú. Cada opción seleccionada por el usuario inicia una tarea en un hilo separado, permitiendo que la interfaz de usuario siga siendo responsiva mientras se procesan las operaciones de entrada/salida en el fondo. Al final del menú, se ofrece una opción para salir, cerrando el bucle y terminando el programa de forma controlada.

```
def main_menu(fs):
    # Presenta un menú continuo que permite al usuario interactuar con el sistema FiUnamFS a través de varias opciones.
    # Utiliza hilos para gestionar las operaciones sin bloquear la interacción del usuario con el menú.
    while True:
        print("\nMenu:")
        print("Nota: El formato para pasar la ruta debe ser C:\\Users\\YourUsername\\Desktop--ComoEjemplo\\Carpeta\\nombreArchivo.terminacion")
        print("1. Listar los contenidos del directorio")
        print("2. Copiar un archivo del FiUnamFS a tu sistema")
        print("3. Copiar un archivo de tu computadora al FiUnamFS")
        print("4. Eliminar un archivo del FiUnamFS")
        print("5. Salir")
        choice = input("Ingrese su elección: ")
```

```python
if choice == '1':
    thread = Thread(target=threaded_task, args=(fs, "list"))
    thread.start()
    thread.join()
elif choice == '2':
    filename = input("Ingrese el nombre del archivo a copiar del FiUnamFS: ")
    destination = input("Ingrese la ruta de destino en su sistema: ")
    thread = Thread(target=threaded_task, args=(fs, "copy_from_fs", filename, destination))
    thread.start()
    thread.join()
elif choice == '3':
    source = input("Ingrese la ruta del archivo en su sistema para copiar al FiUnamFS: ")
    filename = input("Ingrese el nombre bajo el cual guardar el archivo en el FiUnamFS: ")
    thread = Thread(target=threaded_task, args=(fs, "copy_to_fs", source, filename))
    thread.start()
    thread.join()
elif choice == '4':
    filename = input("Ingrese el nombre del archivo a eliminar del FiUnamFS: ")
    thread = Thread(target=threaded_task, args=(fs, "delete", filename))
    thread.start()
    thread.join()
elif choice == '5':
    print("Saliendo...")
    break
else:
    print("Opción no válida. Por favor intente de nuevo.")
```

1. LISTAR CONTENIDOS DEL DIRECTORIO

```python
def list_files(self):
    # Lista los archivos en el directorio principal del FiUnamFS, excluyendo entradas vacías y no utilizadas.
    # Lee las entradas del directorio después de saltar el superbloque.
    with self.lock:
        with open(self.filepath, 'r+b') as f:
            f.seek(1024)  # Posicionarse al inicio del directorio
            entries = []
            for _ in range(64):
                entry = f.read(64)
                filename = entry[1:16].decode('ascii', 'ignore').replace('\x00', ' ').strip()
                if filename and filename != '###############':  # Filtrar entradas vacías y marcadas como no utilizadas
                    entries.append(filename)
    return entries
```

Para lograr el funcionamiento realizamos lo siguiente:

1. Control de Acceso Concurrente: Utilizamos un bloqueo para prevenir conflictos de acceso al archivo del sistema de archivos.

2. Acceso al Archivo:Abrimos el archivo del sistema en modo lectura y escritura binaria y se posiciona al inicio del directorio, justo después del superbloque.

3. Lectura de Entradas: En un bucle, se lee bloques de 64 bytes que representan las entradas de archivo en el directorio.

4. Extracción y Filtrado de Nombres:Se decodifica y limpia los nombres de archivo de caracteres nulos y espacios. Omite entradas vacías o marcadas como no utilizadas.

5. Devolución de Resultados:Finalmente se retorna una lista con los nombres de los archivos válidos encontrados en el directorio.

2. **COPIAR UNO DE LOS ARCHIVOS DE DENTRO DEL FiUnamFS HACIA TU SISTEMA.**

```python
# Copia un archivo desde FiUnamFS al sistema del usuario, buscando por nombre y leyendo su tamaño y posición.
# Es necesario asegurarse que el directorio de destino termine en el separador de carpetas adecuado.
def copy_file_from_fs(fs, filename, destination_dir):
    with fs.lock:
        with open(fs.filepath, 'rb') as disk:
            disk.seek(1024)  # Saltar superbloque
            for _ in range(64):
                entry = disk.read(64)
                file_name = struct.unpack('15s', entry[1:16])[0].decode('ascii').strip('\x00').strip()
                if file_name == filename.strip():
                    file_size = struct.unpack('<I', entry[16:20])[0]
                    start_cluster = struct.unpack('<I', entry[20:24])[0]
                    start_byte = 1024 * start_cluster
                    disk.seek(start_byte)
                    data = disk.read(file_size)

                    # Asegurarse de que el destino_dir termine con un separador de carpetas
                    if not destination_dir.endswith(os.path.sep):
                        destination_dir += os.path.sep

                    # Construir la ruta completa del archivo de destino
                    destination_file_path = destination_dir + filename.strip()
                    with open(destination_file_path, 'wb') as new_file:
                        new_file.write(data)
                    return "Archivo copiado con exito"
            return "Archivo no encontrado"
```

1. Utilización de un Bloqueo de Concurrencia: Se utiliza un bloqueo para garantizar el acceso exclusivo al sistema de archivos durante la operación de copia.

2. Acceso y Posicionamiento en el Archivo: Se abre el archivo del sistema en modo lectura binaria y se posiciona al inicio del directorio de archivos, justo después del superbloque.

3. Búsqueda y Decodificación del Archivo: Se recorren las entradas del directorio, se decodifican los nombres de los archivos y se busca una coincidencia con el nombre del archivo solicitado.

4. Extracción de Información del Archivo: Si se encuentra el archivo, se extrae información crucial como el tamaño del archivo y el cluster inicial donde está almacenado.

5. Lectura de Datos del Archivo: Se calcula la posición exacta del archivo en bytes dentro del disco y se leen los datos del archivo desde esa posición.

6. Preparación del Directorio de Destino: Se verifica y ajusta la ruta de destino para asegurarse de que termine con el separador de carpetas adecuado.

7. Escritura del Archivo en el Destino: Se construye la ruta completa del archivo de destino, se escriben los datos en un nuevo archivo en esa ubicación y se cierra el archivo.

8. Finalización y Reporte de la Operación: Se devuelve un mensaje indicando el éxito de la operación si se copió correctamente el archivo, o un mensaje de error si el archivo no se encontró.

3. COPIAR UN ARCHIVO DE TU COMPUTADORA HACIA TU FiUnamFS

```python
# Busca y retorna el índice del primer cluster libre en el disco.
# Comienza la búsqueda después del superbloque y verifica si cada cluster está vacío comparando con una secuencia de ceros.
def find_free_cluster(disk, num_clusters, cluster_size):
    free_cluster = -1
    disk.seek(cluster_size)  # Saltar el superbloque
    for i in range(1, num_clusters):
        disk.seek(cluster_size * i)
        if disk.read(cluster_size) == b'\x00' * cluster_size:
            free_cluster = i
            break
    return free_cluster
```

Función 'find_free_cluster'

1. Inicio de la Búsqueda: Se salta el superbloque para iniciar la búsqueda de clusters libres desde el primer cluster del sistema.

2. Verificación de Clusters Vacíos: Se itera a través de cada cluster para verificar si está vacío, comparando su contenido con una secuencia de bytes nulos.

3. Identificación del Cluster Libre: Si se encuentra un cluster vacío, se retorna su índice. Si no se encuentra ninguno, se retorna `-1` indicando que no hay clusters libres disponibles.

```python
def copy_file_to_fs(fs, source, filename):
            disk.seek(0)  # Posicionarse al inicio del archivo para leer el superbloque
            superblock = disk.read(1024)  # Leer el superbloque completo

            cluster_size = struct.unpack('<I', superblock[40:44])[0]
            total_clusters = struct.unpack('<I', superblock[50:54])[0]

            disk.seek(1024)  # Posicionarse al inicio del directorio
            empty_entry_index = -1
            for i in range(64):
                entry = disk.read(64)
                file_name = struct.unpack('15s', entry[1:16])[0].decode('ascii').rstrip('\x00').strip()
                print(f"Entry {i}: '{file_name}'")  # Imprime cada nombre de archivo encontrado en el directorio
                if file_name == '##############':  # Usar 14 signos # para estar en línea con los resultados observados
                    empty_entry_index = i
                    break

            if empty_entry_index == -1:
                print("Debug: No space in directory")  # Mensaje de depuración adicional
                return "Sin espacio en el disco"

            free_cluster = find_free_cluster(disk, total_clusters, cluster_size)
            if free_cluster == -1:
                print("Debug: No free cluster available")  # Mensaje de depuración adicional
                return "No hay un cluster libre disponible"

            with open(source, 'rb') as file:
                data = file.read()
                file_size = len(data)

            disk.seek(cluster_size * free_cluster)
            disk.write(data)

            disk.seek(1024 + 64 * empty_entry_index)
            filename_encoded = filename.encode('ascii')
            filename_padded = filename_encoded.ljust(15, b' ')  # Rellenar con espacios si es necesario
            entry_data = struct.pack('<c15sII52s', b'-', filename_padded, file_size, free_cluster, b'\x00' * 52)
            print(f"Writing entry data at index {empty_entry_index}: {entry_data}")  # Debugging output
            disk.seek(1024 + 64 * empty_entry_index)
            disk.write(entry_data)

            return "Archivo copiado exitosamente"
```

Función `copy_file_to_fs`

1. Control de Acceso Concurrente: Se utiliza un bloqueo para garantizar el acceso exclusivo al sistema de archivos durante la operación de copia.

2. Lectura del Superbloque: Se posiciona al inicio del archivo del sistema y se lee el superbloque completo para obtener información sobre el tamaño y la cantidad total de clusters.

3. Búsqueda de Entrada Vacía en el Directorio: Se recorren las entradas del directorio buscando una que esté marcada como no utilizada. Si no se encuentra espacio disponible, se retorna un mensaje indicando falta de espacio.

4. Búsqueda de Cluster Libre: Se utiliza la función 'find_free_cluster' para encontrar un cluster libre donde se pueda almacenar el archivo.

5. Lectura de Datos del Archivo Fuente: Se abre el archivo origen en modo lectura binaria y se leen todos sus datos.

6. Escritura de Datos en el Cluster Libre: Se posiciona al inicio del cluster libre y se escriben los datos del archivo.

7. Actualización del Directorio: Se construye una nueva entrada para el archivo con su nombre, tamaño y la posición del cluster. Luego, se actualiza el directorio con esta nueva entrada.

8. Confirmación de la Operación: Se retorna un mensaje indicando que el archivo fue copiado exitosamente.

4. ELIMINAR UN ARCHIVO DEL FiUnamFS

```python
# Elimina un archivo de FiUnamFS marcando su entrada de directorio como no utilizada.
# Busca el archivo por nombre, y si lo encuentra, reescribe su entrada en el directorio con un patrón que indica borrado.
def delete_file(fs, filename):
    filename = filename.strip()  # Asegúrate de que el nombre del archivo no tenga espacios al inicio ni al final
    with fs.lock:
        with open(fs.filepath, 'r+b') as disk:
            disk.seek(1024)  # Saltar el superbloque
            for i in range(64):
                entry = disk.read(64)
                file_name = entry[1:16].decode('ascii').replace('\x00', '').strip()
                if file_name == filename:
                    disk.seek(1024 + 64 * i)
                    # Escribir exactamente 64 bytes, llenando con ceros después de '/###############'
                    disk.write(b'/###############' + b'\x00' * (64 - 17))  # Asegura exactamente 64 bytes
                    return "Archivo borrado exitosamente"
            return "Archivo no encontrado"
```

1. Preparación del Nombre del Archivo: Se ajusta el nombre del archivo para eliminar espacios al inicio y al final, asegurando que la búsqueda sea precisa.

2. Control de Acceso Concurrente: Se utiliza un bloqueo para garantizar que no haya conflictos de acceso mientras se realiza la operación de eliminación.

3. Posicionamiento en el Directorio: Se accede al archivo del sistema y se posiciona el puntero justo después del superbloque para empezar a buscar en el directorio de archivos.

4. Búsqueda del Archivo: Se recorre cada una de las 64 posibles entradas del directorio, decodificando y comparando los nombres de los archivos con el solicitado.

5. Marcado del Archivo para Borrado: Si se encuentra el archivo, se posiciona el puntero en la entrada correspondiente del directorio y se sobrescribe con un patrón (`'/###############' + b'\x00' * (64 - 17)`) que indica que el archivo ha sido borrado.

6. Confirmación o Reporte de Error: Se devuelve un mensaje indicando que el archivo ha sido borrado exitosamente. Si no se encuentra el archivo, se retorna un mensaje informando que el archivo no fue encontrado.

5. MANEJO DEL HILO

```python
def threaded_task(fs, task, *args):
    if task == "list":
        print("\nListando archivos...")
        print(fs.list_files())
    elif task == "copy_from_fs":
        filename, destination = args
        print(f"\Copiando {filename} desde FS a {destination}...")
        print(copy_file_from_fs(fs, filename, destination))
    elif task == "copy_to_fs":
        source, filename = args
        print(f"\Copiando {source} a FS desde {filename}...")
        print(copy_file_to_fs(fs, source, filename))
    elif task == "delete":
        filename = args[0]
        print(f"\Borrando {filename}...")
        print(delete_file(fs, filename))
```

1. Ejecución de Tareas en un Hilo Separado: Se ejecutan tareas especificadas en un hilo separado para permitir operaciones de I/O sin bloquear la interfaz de usuario principal. Esto mejora la respuesta de la aplicación durante operaciones de archivos intensivos.

2. Listado de Archivos: Si la tarea es "list", se imprime un mensaje indicativo y se muestra la lista de archivos en el sistema FiUnamFS utilizando la función `fs.list_files()`.

3. Copia de Archivos desde el FiUnamFS: Si la tarea es "copy_from_fs", se extraen los argumentos proporcionados (nombre del archivo y directorio de destino), se imprime un mensaje de inicio de copia, y se ejecuta la función `copy_file_from_fs(fs, filename, destination)` para copiar el archivo especificado al directorio de destino.

4. Copia de Archivos hacia el FiUnamFS: Si la tarea es "copy_to_fs", se obtienen los argumentos (ruta del archivo fuente y nombre bajo el cual guardar), se imprime un mensaje de inicio de copia, y se lleva a cabo la copia del archivo desde el sistema del usuario al FiUnamFS utilizando la función `copy_file_to_fs(fs, source, filename)`.

5. Eliminación de Archivos: Si la tarea es "delete", se toma el nombre del archivo desde los argumentos, se imprime un mensaje indicando que el archivo está siendo borrado, y se procede a eliminar el archivo mediante la función `delete_file(fs, filename)`.

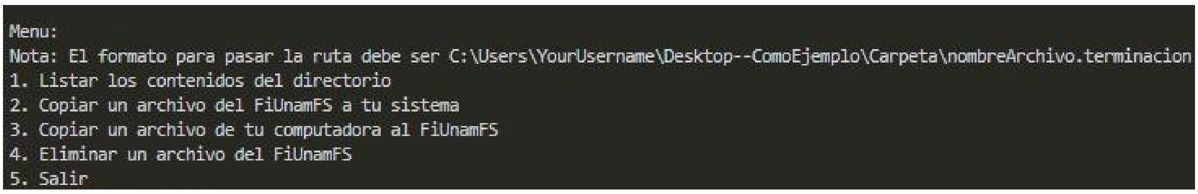
Además, se tiene la opción de Salir para finalizar con la ejecución del programa.

SINCRONIZACIÓN EMPLEADA

En el programa que simula el sistema de archivos FiUnamFS, se utilizó un mecanismo de sincronización basado en **Lock** de la biblioteca threading de Python, para asegurar que las operaciones críticas no interfieran entre sí cuando se ejecutan en múltiples hilos, con lo cual evita las condiciones de carrera que se pudieran presentar, asegurando la consistencia de los datos y manteniendo la integridad del sistema. En el constructor de la clase FiUnamFS, se inicializa un Lock y se utiliza en métodos críticos como **list_files**, **copy_file_from_fs**, **copy_file_to_fs** y **delete_file**. Cada vez que un hilo realiza una operación que puede modificar el sistema de archivos, obtiene el Lock para garantizar que otros hilos no interfieran, para posteriormente liberarlo cuando termina la operación, asegurando que las operaciones concurrentes se manejen de manera ordenada y segura, previniendo posibles eventos en donde se corrompan los datos.

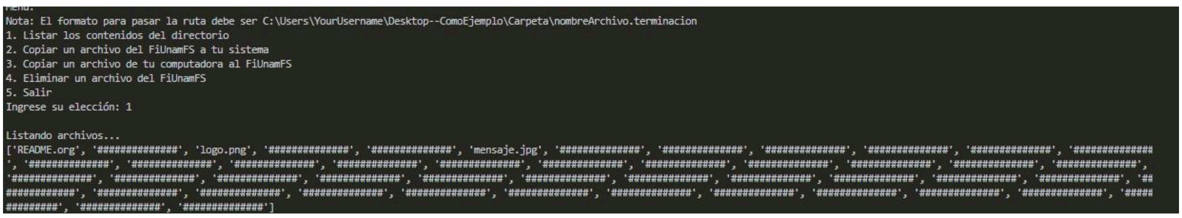
EJEMPLOS DE USO

Primero, corremos el programa, el cual nos mostrará el menú con lo siguiente:



[Menú inicial]

Seleccionando la opción (1) nos muestra lo siguiente:

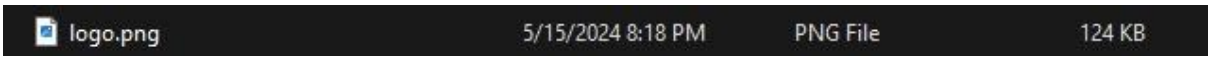


[Listado del contenido del directorio seleccionado]

Seleccionando la opción (2) nos muestra lo siguiente:

```
Menu:
Nota: El formato para pasar la ruta debe ser C:\Users\YourUsername\Desktop--ComoEjemplo\Carpeta\nombreArchivo.terminacion
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS a tu sistema
3. Copiar un archivo de tu computadora al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Ingrese su elección: 2
Ingrese el nombre del archivo a copiar del FiUnamFS: logo.png
Ingrese la ruta de destino en su sistema: C:\Users\AlamLR\Desktop\proyectoSO
\Copiando logo.png desde FS a C:\Users\AlamLR\Desktop\proyectoSO...
Archivo copiado con exito
```

[Creando copia de un archivo de FiUnamFS a nuestro sistema]



[Captura de la copia del archivo generada en nuestro sistema]

Seleccionando la opción (3) nos muestra lo siguiente:

```
Menu:
Nota: El formato para pasar la ruta debe ser C:\Users\YourUsername\Desktop--ComoEjemplo\Carpeta\nombreArchivo.terminaci
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS a tu sistema
3. Copiar un archivo de tu computadora al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Ingrese su elección: 3
Ingrese la ruta del archivo en su sistema para copiar al FiUnamFS: C:\Users\AlamLR\Desktop\proyectoSO\prueba.txt
Ingrese el nombre bajo el cual guardar el archivo en el FiUnamFS: prueba1.txt
\Copiando C:\Users\AlamLR\Desktop\proyectoSO\prueba.txt a FS desde prueba1.txt...
```

[Copiando uno de nuestros archivos a FiUnamFS]

```
Listando archivos...
['README.org', 'prueba1.txt', '##############', '##############', 'mensaje.jpg', '##############', '##############', '##############', '##############', '##############', '##############', '##########
###', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############
', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############',
'##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '#
##########', '##############']
```

[Listado del contenido del directorio para verificar que se realizó la copia]

Seleccionando la opción (4) nos muestra lo siguiente:

```
Menu:
Nota: El formato para pasar la ruta debe ser C:\Users\YourUsername\Desktop--ComoEjemplo\Carpeta\nombreArchivo.terminacion
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS a tu sistema
3. Copiar un archivo de tu computadora al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Ingrese su elección: 4
Ingrese el nombre del archivo a eliminar del FiUnamFS: prueba1.txt
\Borrando prueba1.txt...
Archivo borrado exitosamente

Menu:
Nota: El formato para pasar la ruta debe ser C:\Users\YourUsername\Desktop--ComoEjemplo\Carpeta\nombreArchivo.terminacion
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS a tu sistema
3. Copiar un archivo de tu computadora al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Ingrese su elección: 1

Listando archivos...
['README.org', '##############', '##############', 'mensaje.jpg', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '###
##########', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##
##########', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '#
##########', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############', '##############']
```

[Eliminamos el archivo copiado en la opción 3, además de un listado para verificarlo]

Seleccionando la opción (5) nos muestra lo siguiente:

```
5. Salir
Ingrese su elección: 5
Saliendo...
```

[Saliendo del programa, terminando la ejecución]