UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Semestre 2024-2

Sistemas Operativos

Grupo 06

Proyecto 1

Alumnos
Pozos Hernández Angel
Reynoso Ortega Francisco Javier

Profesor: Gunnar Eyal Wolf Iszaevich

Índice

Planteamiento del Problema:

El proyecto busca desarrollar un sistema de archivos llamado FiUnamFS, diseñado para gestionar archivos dentro de una imagen de disco. Este sistema de archivos tiene como objetivo proporcionar funcionalidades básicas de lectura, escritura, copia y eliminación de archivos, manteniendo un directorio de archivos con información relevante como nombre, tamaño, fechas de creación y modificación, así como la ubicación física en el disco.

Desafíos a Resolver:

1. **Diseño del Sistema de Archivos:** Es necesario diseñar una estructura de datos eficiente que represente el sistema de archivos en una imagen de disco. Esto incluye definir la disposición de los datos en el disco, la estructura del directorio y cómo se almacena la información de cada archivo.
2. **Operaciones Básicas:** Implementar operaciones básicas como la creación, lectura, escritura y eliminación de archivos. Estas operaciones deben ser eficientes y seguras, garantizando la integridad de los datos y la consistencia del sistema.
3. **Gestión del Directorio:** El sistema debe ser capaz de mantener un directorio actualizado con la información de los archivos presentes en la imagen de disco. Esto implica manejar la creación, actualización y eliminación de entradas de directorio de manera adecuada.
4. **Sincronización de Acceso:** Dado que el sistema puede ser accedido concurrentemente por múltiples hilos, es crucial implementar mecanismos de sincronización para evitar condiciones de carrera y garantizar la consistencia de los datos.

Explicación del código:

```
import os
import struct
import threading
import datetime
from os import path
```

- `os`: Proporciona funciones para interactuar con el sistema operativo, como acceder al sistema de archivos.
- `struct`: Permite realizar operaciones de empaquetado y desempaquetado de datos binarios, útil para manipular estructuras de datos a nivel de bytes.
- `threading`: Proporciona clases y funciones para la programación concurrente mediante hilos.
- `datetime`: Permite trabajar con objetos de fecha y hora en Python.
- `from os import path`: Importa la función `path` del módulo `os`, que se utiliza más adelante para trabajar con rutas de archivo.

Constantes y Variables Globales:

```
TAMANO_CLUSTER = 2048
DIRECTORIO_INICIO = TAMANO_CLUSTER
TAMANO_ENTRADA = 64
DIRECTORIO_TAMANO = TAMANO_CLUSTER * 4
ENTRADA_VACIA = b'################'
lock = threading.Lock()
```

- `TAMANO_CLUSTER`: Tamaño de cada "cluster" en bytes. Un cluster es la unidad mínima de almacenamiento en el sistema de archivos.
- `DIRECTORIO_INICIO`: Posición de inicio del directorio en la imagen de disco.
- `TAMANO_ENTRADA`: Tamaño de cada entrada en el directorio.
- `DIRECTORIO_TAMANO`: Tamaño total del directorio en la imagen de disco.
- `ENTRADA_VACIA`: Marca utilizada para indicar que una entrada en el directorio está vacía.
- `lock`: Objeto de bloqueo utilizado para sincronizar el acceso a recursos compartidos entre hilos.

Clase `EntradaArchivo`:

```
class EntradaArchivo:
    def __init__(self, datos):
        self.tipo = datos[0]
        self.nombre = datos[1:16].decode('ascii', 'ignore').rstrip('\x00')
        self.tamano = struct.unpack('<I', datos[16:20])[0]
        self.cluster = struct.unpack('<I', datos[20:24])[0]
        self.fecha_creacion = datos[24:38].decode('ascii', 'ignore')
        self.fecha_modificacion = datos[38:52].decode('ascii', 'ignore')
```

- Representa una entrada de archivo en el directorio.
- `__init__`: Método constructor que inicializa los atributos de la entrada de archivo a partir de los datos proporcionados.

Funciones de Utilidad:

- `leer_directorio`: Lee las entradas del directorio desde la imagen de disco.
- `listar_directorio`: Lista los archivos contenidos en el directorio.
- `copiar_a_fiunamfs`: Copia un archivo del sistema actual al sistema de archivos FiUnamFS.
- `copiar_desde_fiunamfs`: Copia un archivo del sistema de archivos FiUnamFS al sistema actual.
- `verificar_superbloque`: Verifica la validez del superbloque del sistema de archivos.
- `eliminar`: Elimina un archivo del sistema de archivos FiUnamFS.
- `mostrar_menu`: Muestra el menú de opciones para el usuario.
- `principal`: Función principal que gestiona la interacción con el usuario.

Función que lee la dirección del archivo

```
def leer_directorio(archivo):
    """Lee las entradas del directorio desde el archivo del sistema."""
    archivo.seek(DIRECTORIO_INICIO)  # El directorio comienza en el segundo cluster
    entradas = []
    for _ in range(DIRECTORIO_TAMANO // TAMANO_ENTRADA):
        datos = archivo.read(TAMANO_ENTRADA)
        if datos[1:16] != ENTRADA_VACIA:  # Verifica si la entrada no está vacía
            entradas.append(EntradaArchivo(datos))
    return entradas
```

Esta función recibe un archivo y lee las entradas del directorio desde ese archivo. Utiliza una estructura de datos llamada `EntradaArchivo` para representar cada entrada del directorio. Retorna una lista de estas entradas.

```
def listar_directorio(fiunamfs_img):

    """Lista los archivos contenidos en el directorio."""

    with lock:  # Bloquea para evitar conflictos entre hilos

        print("Estado del Lock: Adquirido (Listar Directorio)")

    with open(fiunamfs_img, 'rb') as archivo:

        for entrada in leer_directorio(archivo):

            print(f'Nombre: {entrada.nombre}, Tamaño: {entrada.tamano}, Fecha de
creación: {entrada.fecha_creacion}, Fecha de modificación:
{entrada.fecha_modificacion}')

    with lock:

        print("Estado del Lock: Liberado (Listar Directorio)")
```

Esta función lista los archivos contenidos en el directorio del sistema de archivos FiUnamFS. Utiliza la función `leer_directorio()` para obtener las entradas del directorio desde el archivo de imagen (`fiunamfs_img`) y luego imprime información sobre cada entrada, como el nombre del archivo, su tamaño y fechas de creación y modificación.

```
def copiar_a_fiunamfs(fiunamfs_img, archivo_origen, nombre_destino):
    """Copia un archivo del sistema actual al sistema de archivos FiUnamFS."""
    with lock:
        print("Estado del Lock: Adquirido (Copiar a FiUnamFS)")
    with open(fiunamfs_img, 'r+b') as f:
        tam_origen = os.path.getsize(archivo_origen)
        cluster_libre = 5  # Primer cluster libre
        posicion_entrada_libre = None

        f.seek(DIRECTORIO_INICIO)
        for _ in range(DIRECTORIO_TAMANO // TAMANO_ENTRADA):
            posicion_actual = f.tell()
            entrada = f.read(TAMANO_ENTRADA)
            tipo_archivo = entrada[0:1]
            cluster_ini = struct.unpack('<I', entrada[20:24])[0]

            if tipo_archivo == b'/' and posicion_entrada_libre is None:
                posicion_entrada_libre = posicion_actual  # Encuentra primera
entrada libre

            if cluster_ini >= cluster_libre:
                cluster_libre = cluster_ini + 1  # Encuentra el próximo cluster
libre

        if posicion_entrada_libre is None:
            raise Exception("No hay espacio en el directorio")
        else:
            with open(archivo_origen, 'rb') as archivo_origen_f:
                f.seek(cluster_libre * TAMANO_CLUSTER)
                f.write(archivo_origen_f.read())

            f.seek(posicion_entrada_libre)
            f.write(b'-' + nombre_destino.ljust(15).encode('ascii'))
            f.write(struct.pack('<I', tam_origen))
            f.write(struct.pack('<I', cluster_libre))
            fecha_actual =
datetime.datetime.now().strftime('%Y%m%d%H%M%S').encode('ascii')
            f.write(fecha_actual)  # Fecha de creación
            f.write(fecha_actual)  # Fecha de modificación
```

```
    with lock:
        print("Estado del Lock: Liberado (Copiar a FiUnamFS)")
```

Esta función copia un archivo desde el sistema de archivos actual al sistema de archivos FiUnamFS. Primero busca un espacio libre en el directorio, luego copia el contenido del archivo de origen al archivo de imagen FiUnamFS y actualiza la entrada del directorio con la información del nuevo archivo.

```
def copiar_desde_fiunamfs(archivo, nombre_archivo):
    """Copia un archivo del sistema de archivos FiUnamFS al sistema actual."""
    with lock:
        print("Estado del Lock: Adquirido (Copiar desde FiUnamFS)")
    with open(archivo, 'rb') as archivo:
        for entrada in leer_directorio(archivo):
            if entrada.nombre.strip() == nombre_archivo.strip():
                archivo.seek((entrada.cluster + 1) * TAMANO_CLUSTER)
                datos = archivo.read(entrada.tamano)
                with open(nombre_archivo.strip(), 'wb') as archivo_salida:
                    archivo_salida.write(datos)
                with lock:
                    print("Estado del Lock: Liberado (Copiar desde FiUnamFS)")
                return True
    with lock:
        print("Estado del Lock: Liberado (Copiar desde FiUnamFS)")
    return False
```

Esta función copia un archivo desde el sistema de archivos FiUnamFS al sistema actual. Lee el directorio del archivo FiUnamFS para encontrar el archivo con el nombre especificado, luego copia su contenido al sistema actual.

```
def verificar_superbloque(fiunamfs_img):
    """Verifica la validez del superbloque del sistema de archivos."""
    with open(fiunamfs_img, 'rb') as f:
        nombre_fs = f.read(8).decode('ascii').strip()
        version = f.read(7).decode('ascii').strip().replace('-', '.')
```

```
        print(f"Nombre FS leído: {nombre_fs}, Versión leída: {version}")

        resultado = f"Nombre del sistema de archivos: {nombre_fs}, Versión:
{version}\n"
        if nombre_fs != "FiUnamFS" and version != "24.2":
            resultado += f"¡ERROR! La versión o el sistema de archivos no coinciden
con FiUnamFS versión 24.2. Se esperaba 'FiUnamFS' y '24.2' pero se encontró
'{nombre_fs}' y '{version}'\n"
            return resultado
        else:
            resultado += "Superbloque válido\n"
    return resultado
```

verificar_superbloque(fiunamfs_img): Esta función verifica la validez del superbloque del sistema de archivos FiUnamFS. Lee los primeros bytes del archivo de imagen y verifica si el nombre y la versión del sistema de archivos coinciden con los esperados.

```
def eliminar(nombre_archivo, fiunamfs_img):

    """Elimina un archivo del sistema de archivos FiUnamFS."""

    with lock:

        print("Estado del Lock: Adquirido (Eliminar de FiUnamFS)")

    with open(fiunamfs_img, 'r+b') as archivo:

        archivo.seek(1024)  # Saltar el superbloque

        for _ in range(64):

            entrada = archivo.read(64)

            nombre = entrada[1:16].rstrip(b'\x00').decode('ascii',
errors='ignore').strip()

            if nombre == nombre_archivo:

                archivo.seek(-64, os.SEEK_CUR)
```

```
                archivo.write(b'/###############' + b'\x00' * (64 - 17))  # Marca la
entrada como vacía

                print("Archivo borrado exitosamente")

                return

    print("Archivo no encontrado en el directorio")
```

eliminar(nombre_archivo, fiunamfs_img): Esta función elimina un archivo del sistema de archivos FiUnamFS. Busca el archivo en el directorio y marca su entrada como vacía.

```
def mostrar_menu():

    """Muestra el menú de opciones para el usuario."""

    print("1. Listar los contenidos del directorio")

    print("2. Copiar un archivo del FiUnamFS al sistema")

    print("3. Copiar un archivo del sistema al FiUnamFS")

    print("4. Eliminar un archivo del FiUnamFS")

    print("5. Salir")
```

mostrar_menu(): Esta función simplemente muestra un menú de opciones para que el usuario elija qué acción realizar

```
def principal():

    """Función principal que gestiona la interacción con el usuario."""

    fiunamfs_img = 'fiunamfs.img'

    resultado_verificacion = verificar_superbloque(fiunamfs_img)

    print(resultado_verificacion)

    if "¡ERROR!" in resultado_verificacion:

        return  # Termina el programa si la verificación falla



    while True:

        mostrar_menu()

        opcion = input("Elige una opción: ")

        if opcion == '1':

            hilo1 = threading.Thread(target=listar_directorio, args=(fiunamfs_img,))

            hilo1.start()

            hilo1.join()

        elif opcion == '2':

            nombre_archivo = input("Introduce el nombre del archivo a copiar: ")
```

```
            resultado_copia = copiar_desde_fiunamfs(fiunamfs_img, nombre_archivo)

            if resultado_copia:

                print(f'El archivo {nombre_archivo} se copió exitosamente.')

            else:

                print(f'El archivo {nombre_archivo} no fue encontrado en el
FiUnamFS.')

        elif opcion == '3':

            nombre_archivo_origen = input("Introduce el nombre del archivo a copiar
desde el sistema: ")

            nombre_archivo_destino = input("Introduce el nombre del archivo en
FiUnamFS: ")

            try:

                copiar_a_fiunamfs(fiunamfs_img, nombre_archivo_origen,
nombre_archivo_destino)

                print(f'Archivo {nombre_archivo_origen} copiado exitosamente como
{nombre_archivo_destino} en FiUnamFS.')

            except Exception as e:

                print(f'Error al copiar archivo: {e}')

        elif opcion == '4':

            nombre_archivo = input("Introduce el nombre del archivo a eliminar: ")

            eliminar(nombre_archivo, fiunamfs_img)

        elif opcion == '5':
```

```
            break

        else:

            print("Opción no válida. Por favor, elige una opción del menú.")



if __name__ == '__main__':

    principal()
```

principal(): Esta es la función principal del programa. Primero verifica la validez del superbloque del sistema de archivos FiUnamFS. Luego entra en un bucle donde muestra el menú y espera que el usuario elija una opción. Dependiendo de la opción elegida, llama a las funciones correspondientes.

Nota: para que funcione el proyecto el archivo 'fiunamfs' tiene que estar en el mismo directorio que proyecto1.py

. Funcionamiento del código: Ejemplo de ejecución

Al iniciar la ejecución del programa, se abre la siguiente ventana:

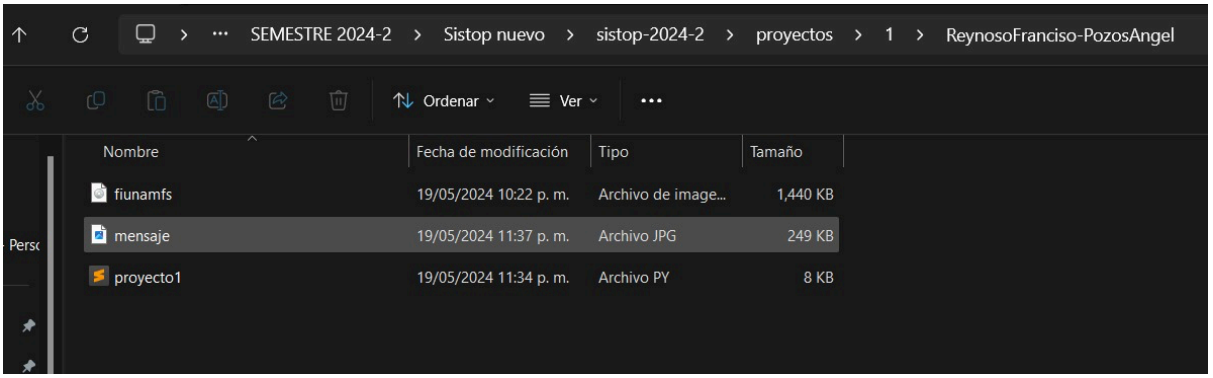
En donde se nos muestran las operaciones disponibles que puede realizar nuestro programa, seleccionaremos la opción 1

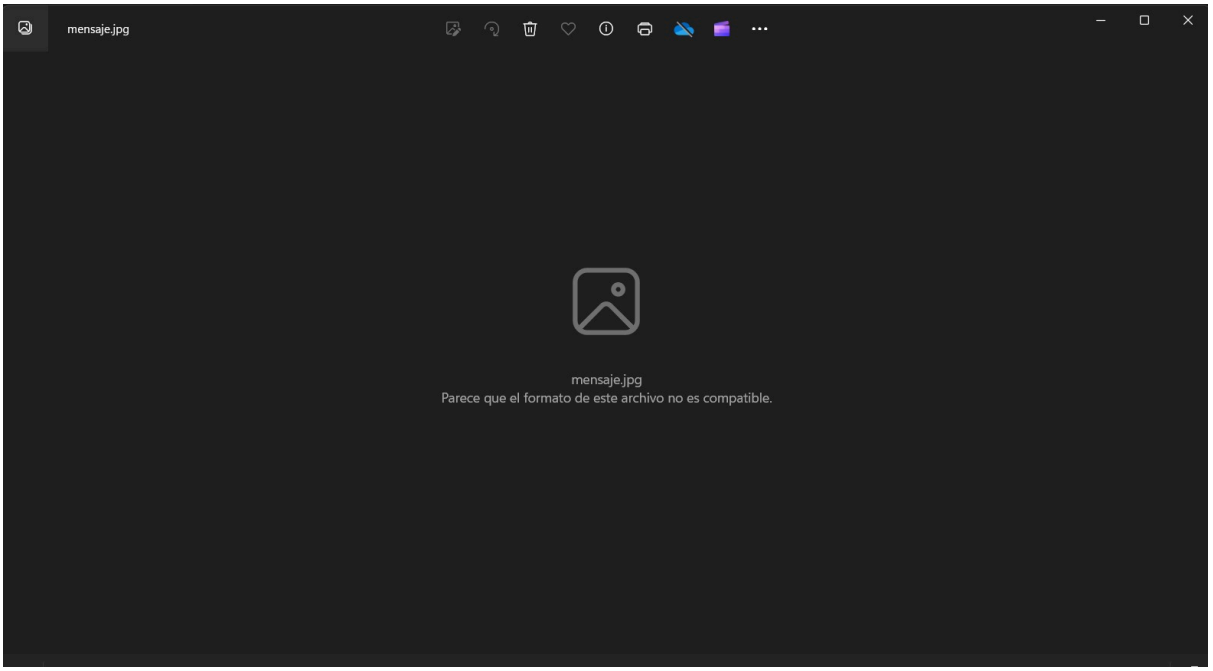
```
Elige una opción: 1
Estado del Lock: Adquirido (Listar Directorio)
Nombre: README.org     , Tamaño: 31222, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: logo.png       , Tamaño: 126423, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: mensaje.jpg    , Tamaño: 254484, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
```

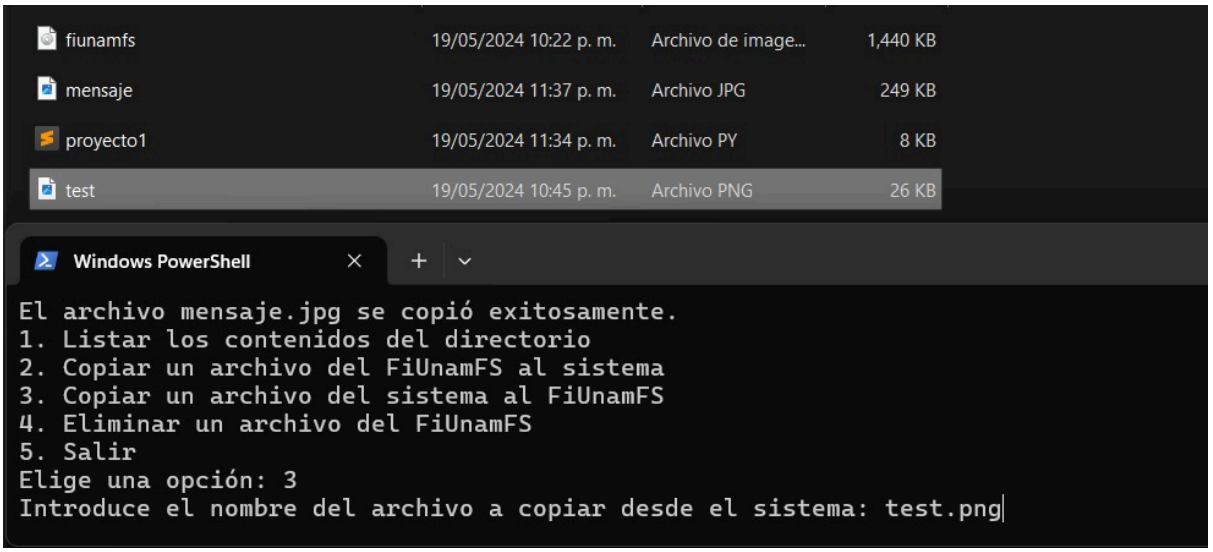
Luego seleccionamos la opción 2

```
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Estado del Lock: Liberado (Listar Directorio)
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS al sistema
3. Copiar un archivo del sistema al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Elige una opción: 2
Introduce el nombre del archivo a copiar: mensaje.jpg
Estado del Lock: Adquirido (Copiar desde FiUnamFS)
Estado del Lock: Liberado (Copiar desde FiUnamFS)
El archivo mensaje.jpg se copió exitosamente.
```

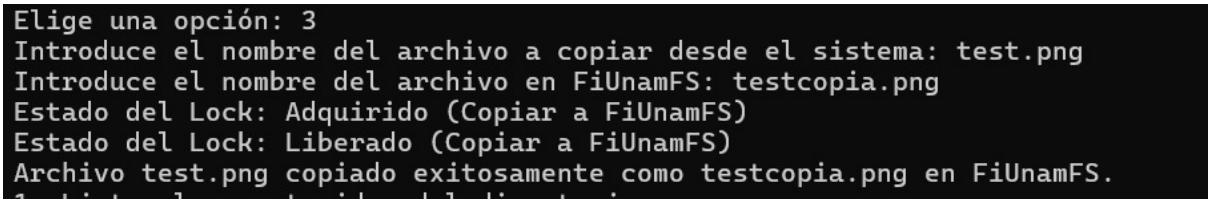
Luego se muestra donde se copio







aquí se copia de nuestro sistema al sistema fiunam



acá se verifica que si se copió

```
Elige una opción: 1
Estado del Lock: Adquirido (Listar Directorio)
Nombre: README.org    , Tamaño: 31222, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: testcopia.png  , Tamaño: 25994, Fecha de creación: 20240519234041, Fecha de modificación: 20240519234041
Nombre: logo.png      , Tamaño: 126423, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: mensaje.jpg   , Tamaño: 254484, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
```

Borramos el archivo

```
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS al sistema
3. Copiar un archivo del sistema al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Elige una opción: 4
Introduce el nombre del archivo a eliminar: testcopia.png
Estado del Lock: Adquirido (Eliminar de FiUnamFS)
Archivo borrado exitosamente
1. Listar los contenidos del directorio
```

Se verifica que se borro

```
Elige una opción: 4
Introduce el nombre del archivo a eliminar: testcopia.png
Estado del Lock: Adquirido (Eliminar de FiUnamFS)
Archivo borrado exitosamente
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS al sistema
3. Copiar un archivo del sistema al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Elige una opción: 1
Estado del Lock: Adquirido (Listar Directorio)
Nombre: README.org    , Tamaño: 31222, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: logo.png      , Tamaño: 126423, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: mensaje.jpg   , Tamaño: 254484, Fecha de creación: 20240508131756, Fecha de modificación: 20240508131756
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
Nombre: ###############, Tamaño: 0, Fecha de creación: 00000000000000, Fecha de modificación: 00000000000000
```

vemos la salida

```
Estado del Lock: Liberado (Listar Directorio)
1. Listar los contenidos del directorio
2. Copiar un archivo del FiUnamFS al sistema
3. Copiar un archivo del sistema al FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Salir
Elige una opción: 5
PS D:\Documentos  (D)\Ing en compu\SEMESTRE 2024-2\Sistop nuevo\sistop-2024-2\proyectos\1\ReynosoFranciso-PozosAngel>
```

CONCLUSIONES

Pozos Hernández Angel

Este proyecto ha sido una inmersión total en el mundo de los sistemas de archivos. Desde la investigación inicial sobre FiUnamFS hasta la implementación de cada función, hemos profundizado significativamente en conceptos como la estructura del directorio, la asignación de clusters y la gestión de archivos. Esta experiencia nos ha brindado una comprensión sólida de cómo los sistemas operativos manejan el almacenamiento y la organización de datos, lo que sin duda será invaluable en nuestro futuro académico y profesional en el campo de la informática.

Reynoso Ortega Francisco Javier

La realización de este proyecto nos ha enseñado la importancia de la colaboración y la resolución de problemas en equipo. Desde la fase inicial de diseño hasta la depuración y optimización del código, hemos enfrentado desafíos que requerían una combinación de creatividad y rigor técnico. A lo largo del proceso, hemos aprendido a comunicarnos de manera efectiva, a dividir tareas de manera equitativa y a apoyarnos mutuamente para superar obstáculos. Esta experiencia nos ha dejado no solo con un proyecto exitoso, sino también con habilidades y herramientas para enfrentar proyectos futuros de manera más efectiva y colaborativa.

Bibliografía:

Libros:

- **"Programación concurrente con Python: Patrones y prácticas para la programación multihilo y asincrónica"**por Bill Scopatz y Gregory Petrić: Este libro ofrece una introducción completa a la programación concurrente en Python, cubriendo temas como hilos, procesos, programación asincrónica y comunicación entre hilos.https://www.codeauni.com/comunidad/blog/56/
- **"Python para la programación de sistemas"** por Dave Beazley: Este libro cubre temas avanzados de programación en Python, incluyendo la programación de sistemas, el desarrollo de redes y la creación de interfaces gráficas.https://www.amazon.com.mx/Python-Para-Principiantes-Aprender-programar/dp/B0BRH4KG4F

Artículos:

- **"Hilos en Python: Una guía para principiantes"** por Real Python: Este artículo proporciona una introducción básica a la programación multihilo en Python, incluyendo la creación de hilos, la sincronización de hilos y la resolución de problemas comunes.https://realpython.com/
- **"Multiprocesamiento en Python: Aprovechando al máximo tu CPU"** por Real Python: Este artículo explica cómo usar el módulo `multiprocessing` de Python para crear procesos paralelos, lo que puede ser útil para tareas

computacionalmente intensivas.https://www.genbeta.com/desarrollo/multiprocesamiento-en-python-benchmarking

Recursos en línea:

- **Documentación oficial de Python sobre hilos:** La documentación oficial de Python proporciona una referencia detallada sobre el módulo `threading` y otros temas relacionados con la programación multihilo.https://python.readthedocs.io/en/v2.7.2/library/threading.html
- **Blog de Pythonistas:** El blog de Pythonistas ofrece una gran cantidad de artículos y tutoriales sobre programación en Python, incluyendo temas relacionados con la concurrencia y el paralelismo.https://pythonistaplanet.com/pythonista/

Vídeos:

- **"Hilos en Python - Explicación Completa"** por Jose Caceres: Este vídeo proporciona una explicación completa de la programación multihilo en Python, cubriendo temas como la creación de hilos, la sincronización de hilos y la resolución de problemas comunes.https://m.youtube.com/watch?v=3Rlh6uUuQqA
- **"Concurrencia y Paralelismo en Python: Multithreading vs Multiprocessing vs Async"** por PyAr - Python Argentina: Este vídeo compara y contrasta diferentes técnicas de concurrencia en Python, como hilos, procesos y programación asincrónica.https://www.youtube.com/watch?v=u77Az26bFPA

Recursos adicionales:

- **Canal de YouTube de Gunnar Wolf:** Gunnar Wolf es un profesor y escritor que ofrece una gran cantidad de vídeos educativos sobre sistemas operativos y programación en general.https://www.youtube.com/c/GunnarWolf
- **Canal de YouTube de CodigoFacilito:** CodigoFacilito es un canal de YouTube que ofrece tutoriales y cursos sobre programación en varios lenguajes, incluyendo Python.https://www.youtube.com/c/codigofacilito