UNIVERSIDAD NACIONAL AUTÓNOMA DE MEXICO

FACULTAD DE INGENIERIA

SISTEMAS OPERATIVOS

PROYECTO #2

(Micro) sistema de archivos multihilos

(Documentación del Proyecto FiUnamFS)

Grupo : 6

Integrantes:

González Iniestra Emilio

Suarez Guzmán Dayna Yarelly

Profesor: Gunnar Eyal Wolf Iszaevich

Semestre 2025-1

Fecha entrega:

07-11-2024

Indice

1. Autores

- **Nombres de los autores**: González Iniestra Emilio y Suarez Guzman Dayna Yarelly
- **Fecha de entrega**: 07/11/2024

2. Descripción General

El proyecto consiste en el desarrollo de un gestor para el sistema de archivos FiUnamFS, utilizado en un entorno académico. Este gestor permite realizar operaciones esenciales, como listar contenidos, copiar archivos desde y hacia el sistema de archivos, y eliminar archivos. Además, el programa incluye la implementación de hilos para manejar operaciones concurrentes y la sincronización entre estos.

3. Requisitos y Entorno

- **Lenguaje de programación**: Python 3.6 o superior

- **Módulos necesarios**:
    - struct: Para manejar la manipulación de datos binarios.
    - os: Para funciones de sistema y manejo de archivos.
    - time: Para obtener y formatear timestamps.
    - threading: Para crear y manejar hilos.
    - tkinter: Para construir la interfaz gráfica de usuario.
- **Entorno de ejecución**: Windows, macOS o Linux con Python 3.6+.

4. Funcionalidades del Programa

El programa desarrollado permite:

1. **Listar los contenidos del directorio**: Muestra los archivos almacenados en FiUnamFS, incluyendo detalles como el tamaño, la fecha de creación y la última modificación.
2. **Copiar un archivo de FiUnamFS al sistema local**: Permite al usuario seleccionar un archivo y copiarlo al sistema de archivos local.
3. **Copiar un archivo del sistema local a FiUnamFS**: Copia un archivo desde la computadora del usuario al sistema de archivos FiUnamFS, verificando el espacio disponible.
4. **Eliminar un archivo de FiUnamFS**: Elimina un archivo del sistema de archivos.
5. **Sincronización y uso de hilos**: El programa utiliza hilos para ejecutar las operaciones de E/S de forma concurrente y evitar el bloqueo de la interfaz gráfica.

5. Explicación del Código

Importación de Módulos

```
1   import struct
2   import os
3   import time
4   import threading
5   import tkinter as tk
6   from tkinter import filedialog, messagebox, Listbox
```

Descripción: Se importan los módulos necesarios:

- struct: Para manipulación de datos binarios en formato little endian, esencial para leer y escribir en el sistema de archivos.
- os: Para operaciones de manejo de archivos y rutas.
- time: Para trabajar con fechas y formatearlas en el formato requerido (AAAAMMDDHHMMSS).
- threading: Para crear y manejar hilos que ejecuten operaciones en paralelo.
- tkinter: Para crear la interfaz gráfica de usuario (GUI).

- filedialog, messagebox, Listbox: Componentes de tkinter para interacciones de usuario, mostrar mensajes y listas.

Parámetros del Sistema de Archivos

```python
# Parámetros del sistema de archivos FiUnamFS
disk_file = "fiunamfs.img"
DISK_SIZE = 1440 * 1024  # Tamaño total del "disco" en bytes
CLUSTER_SIZE = 256 * 4   # Tamaño de cada cluster en bytes
DIRECTORY_START_CLUSTER = 1  # Cluster de inicio del directorio
DIRECTORY_END_CLUSTER = 4    # Cluster final del directorio
DIRECTORY_ENTRY_SIZE = 64    # Tamaño de cada entrada del directorio
FILE_NAME_SIZE = 15          # Tamaño máximo del nombre de archivo
IDENTIFICATION = b"FiUnamFS" # Identificación del sistema de archivos
VERSION = b"25-1"            # Versión del sistema de archivos
```

Descripción: Estas constantes definen los parámetros de FiUnamFS, como el tamaño del "disco", la estructura de clusters y la identificación del sistema.

Validación y Lectura del Superbloque

```python
def validate_and_read_superblock():
    """
    Valida la identificación y versión en el superbloque,
    y lee la etiqueta, tamaño y clusters.
    """
    try:
        with open(disk_file, "rb") as f:
            # Leer y validar Identificación
            f.seek(0)
            ident = f.read(8).strip(b"\x00").decode("ascii")  # Eliminar rellenos nulos
            if ident != IDENTIFICATION.decode("ascii"):
                messagebox.showerror("Error", "Identificación de sistema de archivos no coincide. Operación cancelada.")
                return None

            # Leer y validar Versión
            f.seek(10)
            version = f.read(5).strip(b"\x00").decode("ascii")
            if version != VERSION.decode("ascii"):
                messagebox.showerror("Error", "Versión de sistema de archivos no coincide. Operación cancelada.")
                return None

            # Leer Etiqueta del Volumen
            f.seek(20)
            label = f.read(16).decode("ascii").strip()

            # Leer Tamaño de Cluster
            f.seek(40)
            cluster_size = struct.unpack("<I", f.read(4))[0]

            # Leer Número de Clusters en el Directorio
            f.seek(45)
            directory_clusters = struct.unpack("<I", f.read(4))[0]

            # Leer Número de Clusters de la Unidad Completa
            f.seek(50)
            total_clusters = struct.unpack("<I", f.read(4))[0]

            return {
                "label": label,
                "cluster_size": cluster_size,
                "directory_clusters": directory_clusters,
                "total_clusters": total_clusters
            }
    except FileNotFoundError:
        messagebox.showerror("Error", "El archivo fiunamfs.img no existe.")
        return None
```

Descripción: Esta función valida que el archivo fiunamfs.img sea un sistema de archivos válido y devuelve información importante como la etiqueta del volumen y el tamaño de clusters. Si la identificación o la versión no coinciden, se muestra un mensaje de error y la función retorna None.

Listar los Contenidos del Directorio

```python
def list_directory():
    """
    Lista los archivos en el directorio de FiUnamFS.
    """
    files = []
    superblock = validate_and_read_superblock()
    if not superblock:
        return files

    try:
        with open(disk_file, "rb") as f:
            # Recorrer las entradas del directorio
            for cluster in range(DIRECTORY_START_CLUSTER, DIRECTORY_END_CLUSTER + 1):
                f.seek(cluster * CLUSTER_SIZE)
                for _ in range(CLUSTER_SIZE // DIRECTORY_ENTRY_SIZE):
                    entry_data = f.read(DIRECTORY_ENTRY_SIZE)
                    if not entry_data or entry_data[0] == 0x23:  # Entrada vacía '#'
                        continue
                    if entry_data[0] != 0x2e:  # Verificar tipo de archivo '.'
                        continue
                    # Obtener nombre del archivo
                    filename = entry_data[1:16].decode("ascii").strip("-")
                    # Obtener tamaño del archivo
                    filesize = struct.unpack("<I", entry_data[16:20])[0]
                    # Obtener fechas de creación y modificación
                    created_at = entry_data[24:38].decode("ascii")
                    modified_at = entry_data[38:52].decode("ascii")
                    files.append(f"{filename} - {filesize} bytes - Creado: {created_at} - Modificado: {modified_at}")
    except FileNotFoundError:
        messagebox.showerror("Error", "El archivo fiunamfs.img no existe.")
    return files
```

Descripción: Recorre las entradas del directorio en los clusters correspondientes y recopila información sobre los archivos, como el nombre, el tamaño y las fechas de creación y modificación. Las entradas vacías (marcadas con #) se omiten.

Copiar un Archivo de FiUnamFS al Sistema Local

```python
def copy_to_local(file_name):
    """
    Inicia un hilo para copiar un archivo de FiUnamFS al sistema local.
    """
    # Iniciar hilo de copia
    threading.Thread(target=copy_to_local_thread, args=(file_name,)).start()

def copy_to_local_thread(file_name):
    """
    Hilo que copia un archivo de FiUnamFS al sistema local.
    """
    save_path = filedialog.asksaveasfilename(title="Guardar archivo en sistema local", initialfile=file_name)
    if not save_path:
        return

    with threading.Lock():
        with open(disk_file, "rb") as f:
            found = False
            for cluster in range(DIRECTORY_START_CLUSTER, DIRECTORY_END_CLUSTER + 1):
                f.seek(cluster * CLUSTER_SIZE)
                for _ in range(CLUSTER_SIZE // DIRECTORY_ENTRY_SIZE):
                    entry_data = f.read(DIRECTORY_ENTRY_SIZE)
                    entry_name = entry_data[1:16].decode("ascii").strip("-")
                    if entry_name == file_name:
                        filesize = struct.unpack("<I", entry_data[16:20])[0]
                        start_cluster = struct.unpack("<I", entry_data[20:24])[0]
                        f.seek(start_cluster * CLUSTER_SIZE)
                        data = f.read(filesize)
                        with open(save_path, "wb") as out_file:
                            out_file.write(data)
                        operation_status['message'] = f"Archivo '{file_name}' copiado a '{save_path}'"
                        operation_status['success'] = True
                        found = True
                        break
                if found:
                    break
            else:
                operation_status['message'] = f"Archivo '{file_name}' no encontrado en FiUnamFS."
                operation_status['success'] = False

    # Notificar al hilo principal
    operation_event.set()
```

Descripción: copy_to_local inicia un hilo que ejecuta copy_to_local_thread, la cual busca el archivo en FiUnamFS y lo copia al sistema local. Se utiliza un threading.Lock para asegurar que la operación sea segura cuando se accede al archivo compartido.

Copiar un Archivo del Sistema Local a FiUnamFS

```
148    def copy_to_fiunamfs():
149        """
150        Inicia un hilo para copiar un archivo desde el sistema local a FiUnamFS.
151        """
152        threading.Thread(target=copy_to_fiunamfs_thread).start()
153
154    def copy_to_fiunamfs_thread():
155        """
156        Hilo que copia un archivo desde el sistema local a FiUnamFS.
157        """
158        local_path = filedialog.askopenfilename(title="Selecciona un archivo para copiar a FiUnamFS")
159        if not local_path:
160            return
161
162        file_name = os.path.basename(local_path)
163        file_size = os.path.getsize(local_path)
164
165        # Tamaño total disponible para archivos en FiUnamFS (excluyendo superbloque y directorio)
166        data_space_start = (DIRECTORY_END_CLUSTER + 1) * CLUSTER_SIZE
167        max_data_space = DISK_SIZE - data_space_start
168
169        # Verificar espacio disponible
170        total_used_space = get_total_used_space()
171        available_space = max_data_space - total_used_space
172
173        if file_size > available_space:
174            operation_status['message'] = f"No hay suficiente espacio en FiUnamFS para copiar '{file_name}'."
175            operation_status['success'] = False
176            operation_event.set()
177            return
178
179        with threading.Lock():
180            with open(disk_file, "r+b") as f, open(local_path, "rb") as in_file:
181                # Encontrar un espacio vacío en el directorio
182                found_space = False
183                for cluster in range(DIRECTORY_START_CLUSTER, DIRECTORY_END_CLUSTER + 1):
184                    f.seek(cluster * CLUSTER_SIZE)
185                    for _ in range(CLUSTER_SIZE // DIRECTORY_ENTRY_SIZE):
```

```
185                    for _ in range(CLUSTER_SIZE // DIRECTORY_ENTRY_SIZE):
186                        pos = f.tell()
187                        entry_data = f.read(DIRECTORY_ENTRY_SIZE)
188                        if entry_data[0] == 0x23:  # Entrada vacía '#'
189                            # Escribir entrada de directorio
190                            f.seek(pos)
191                            f.write(b".")
192                            f.write(file_name.encode("ascii").ljust(FILE_NAME_SIZE, b"-"))
193                            f.write(struct.pack("<I", file_size))
194                            # Calcular cluster inicial
195                            start_cluster = (DISK_SIZE - available_space) // CLUSTER_SIZE
196                            f.write(struct.pack("<I", start_cluster))
197                            # Escribir fechas de creación y modificación
198                            timestamp = time.strftime("%Y%m%d%H%M%S").encode("ascii")
199                            f.write(timestamp)  # Fecha de creación
200                            f.write(timestamp)  # Fecha de modificación
201                            # Rellenar espacio restante de la entrada
202                            f.write(b"\x00" * (DIRECTORY_ENTRY_SIZE - (f.tell() - pos)))
203                            # Escribir datos del archivo
204                            f.seek(start_cluster * CLUSTER_SIZE)
205                            f.write(in_file.read())
206                            operation_status['message'] = f"Archivo '{file_name}' copiado al sistema de archivos."
207                            operation_status['success'] = True
208                            found_space = True
209                            break
210                    if found_space:
211                        break
212                else:
213                    operation_status['message'] = "No se encontró espacio libre en el directorio de FiUnamFS."
214                    operation_status['success'] = False
215
216        # Notificar al hilo principal
217        operation_event.set()
```

Descripción: Esta función busca un espacio disponible en el directorio y copia un archivo desde el sistema local a FiUnamFS. Si no hay suficiente espacio, muestra un mensaje de error. La operación se realiza en un hilo para evitar bloquear la interfaz gráfica.

Eliminar un Archivo de FiUnamFS

```
219  def delete_file(file_name):
220      """
221      Inicia un hilo para eliminar un archivo del FiUnamFS.
222      """
223      threading.Thread(target=delete_file_thread, args=(file_name,)).start()
224
225  def delete_file_thread(file_name):
226      """
227      Hilo que elimina un archivo del FiUnamFS.
228      """
229      with threading.Lock():
230          with open(disk_file, "r+b") as f:
231              found = False
232              for cluster in range(DIRECTORY_START_CLUSTER, DIRECTORY_END_CLUSTER + 1):
233                  f.seek(cluster * CLUSTER_SIZE)
234                  for _ in range(CLUSTER_SIZE // DIRECTORY_ENTRY_SIZE):
235                      pos = f.tell()
236                      entry_data = f.read(DIRECTORY_ENTRY_SIZE)
237                      entry_name = entry_data[1:16].decode("ascii").strip("-")
238                      if entry_name == file_name:
239                          # Marcar entrada como vacía
240                          f.seek(pos)
241                          f.write(b"#" + b"-" * (DIRECTORY_ENTRY_SIZE - 1))
242                          operation_status['message'] = f"Archivo '{file_name}' eliminado del sistema de archivos."
243                          operation_status['success'] = True
244                          found = True
245                          break
246                  if found:
247                      break
248              else:
249                  operation_status['message'] = f"Archivo '{file_name}' no encontrado en FiUnamFS."
250                  operation_status['success'] = False
251
252      # Notificar al hilo principal
253      operation_event.set()
```

Descripción: delete_file inicia un hilo que ejecuta delete_file_thread, la cual marca la entrada del archivo como vacía, eliminándolo del sistema de archivos. La operación se sincroniza con un threading.Lock.

Cálculo del Espacio Total Usado

```
255  def get_total_used_space():
256      """
257      Calcula el espacio total ocupado en FiUnamFS, excluyendo el superbloque y directorio.
258      """
259      total_used = 0
260      try:
261          with open(disk_file, "rb") as f:
262              # Recorrer las entradas del directorio
263              for cluster in range(DIRECTORY_START_CLUSTER, DIRECTORY_END_CLUSTER + 1):
264                  f.seek(cluster * CLUSTER_SIZE)
265                  for _ in range(CLUSTER_SIZE // DIRECTORY_ENTRY_SIZE):
266                      entry_data = f.read(DIRECTORY_ENTRY_SIZE)
267                      if entry_data and entry_data[0] == 0x2e:  # Entrada ocupada '.'
268                          filesize = struct.unpack("<I", entry_data[16:20])[0]
269                          total_used += filesize
270      except FileNotFoundError:
271          messagebox.showerror("Error", "El archivo fiunamfs.img no existe.")
272      return total_used
```

Descripción: Calcula el espacio total usado en FiUnamFS recorriendo las entradas ocupadas en el directorio y sumando el tamaño de los archivos.

Interfaz Gráfica y Manejo de Estado

```
274  def refresh_list():
275      """
276      Actualiza la lista de archivos en la interfaz gráfica.
277      """
278      file_list.delete(0, tk.END)
279      files = list_directory()
280      for file in files:
281          file_list.insert(tk.END, file)
282
283  def check_operation_status():
284      """
285      Verifica el estado de las operaciones realizadas por los hilos y actualiza la interfaz.
286      """
287      if operation_event.is_set():
288          if operation_status['success']:
289              messagebox.showinfo("Éxito", operation_status['message'])
290          else:
291              messagebox.showerror("Error", operation_status['message'])
292          # Limpiar el estado para futuras operaciones
293          operation_event.clear()
294          operation_status['message'] = ''
295          operation_status['success'] = False
296          # Refrescar la lista de archivos
297          refresh_list()
298      # Continuar verificando periódicamente
299      root.after(100, check_operation_status)
300
301  # Configuración de la interfaz gráfica
302  root = tk.Tk()
303  root.title("Gestor de FiUnamFS")
304
305  # Lista de archivos en la interfaz
306  file_list = Listbox(root, width=80)
307  file_list.pack()
```

```
309  # Botones de operación
310  btn_list = tk.Button(root, text="Listar Archivos", command=refresh_list)
311  btn_list.pack(pady=5)
312
313  btn_copy_to_local = tk.Button(root, text="Copiar a Local",
314                                command=lambda: copy_to_local(file_list.get(tk.ACTIVE).split(" - ")[0]))
315  btn_copy_to_local.pack(pady=5)
316
317  btn_copy_to_fiunamfs = tk.Button(root, text="Copiar a FiUnamFS", command=copy_to_fiunamfs)
318  btn_copy_to_fiunamfs.pack(pady=5)
319
320  btn_delete = tk.Button(root, text="Eliminar Archivo",
321                         command=lambda: delete_file(file_list.get(tk.ACTIVE).split(" - ")[0]))
322  btn_delete.pack(pady=5)
323
324  # Iniciar la verificación del estado de operaciones
325  root.after(100, check_operation_status)
326
327  # Ejecutar la aplicación
328  root.mainloop()
```

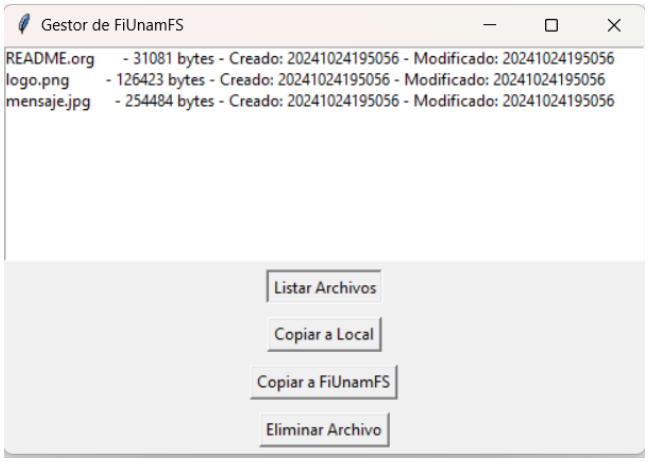
Descripción: La interfaz gráfica se configura con tkinter. Los botones permiten al usuario realizar las operaciones de listar, copiar y eliminar archivos. La función check_operation_status monitorea el estado de las operaciones realizadas en los hilos y muestra mensajes de éxito o error.

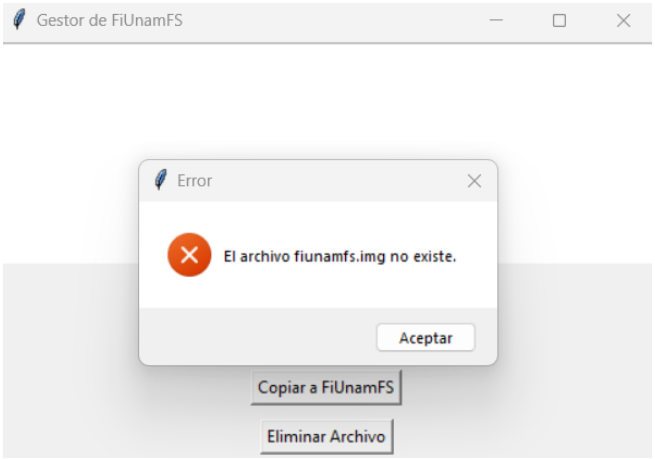
6. Estrategia de Implementación y Sincronización

El programa utiliza hilos para manejar operaciones de E/S sin bloquear la interfaz, proporcionando una experiencia fluida para el usuario. El threading.Lock asegura que las operaciones críticas (como escribir o leer del archivo fiunamfs.img) no interfieran entre sí.

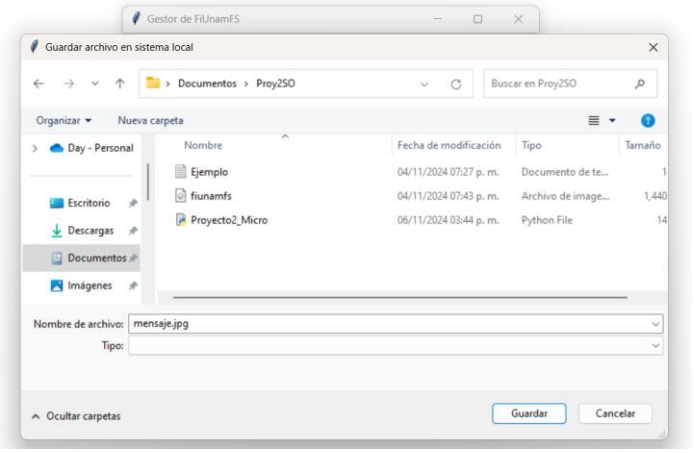
7. Ejemplos de Uso

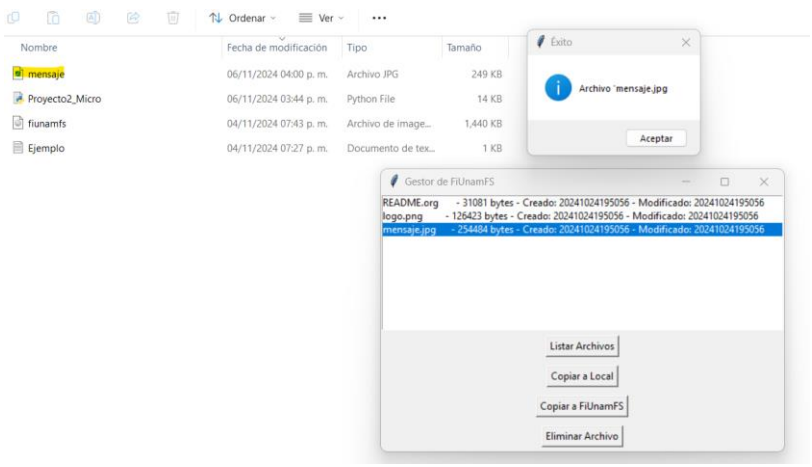
Casos detallados

- **Listar Archivos en FiUnamFS**
    - **Acción**: Presiona el botón "Listar Archivos".
    - **Resultado esperado**: Aparecerá una lista de archivos en la interfaz, mostrando el nombre, el tamaño, la fecha de creación y la última modificación.
    - 

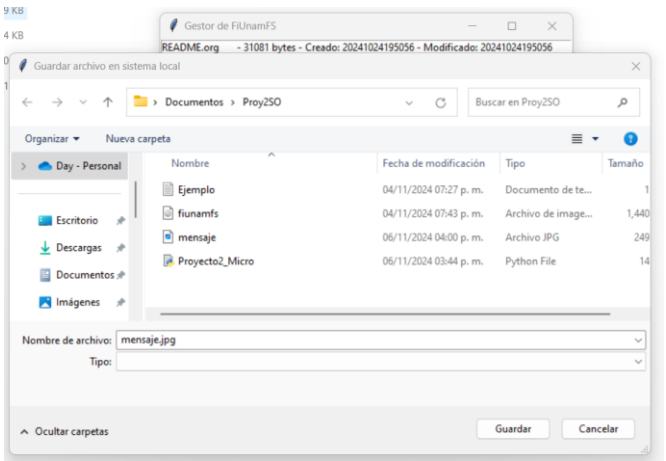
    - **Caso de error**: Si el archivo fiunamfs.img no existe o no es un sistema de archivos válido, se mostrará un mensaje de error.
    - 

- **Copiar un Archivo de FiUnamFS al Sistema Local**
    - **Acción**: Selecciona un archivo de la lista y presiona el botón "Copiar a Local".
    - **Resultado esperado**: Se abrirá un cuadro de diálogo para guardar el archivo en el sistema local. El archivo se copiará y se mostrará un mensaje de éxito.

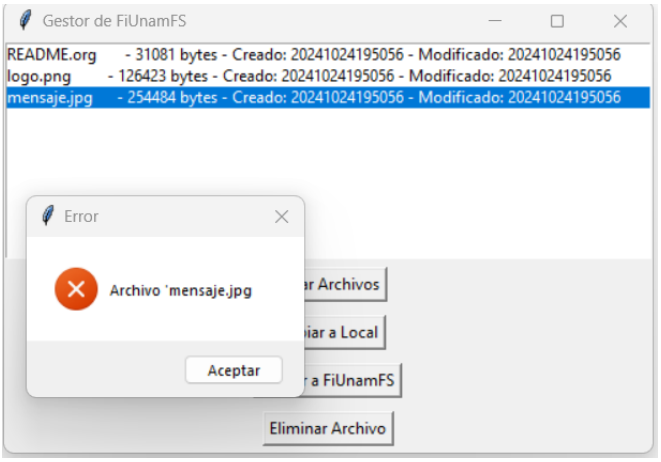




- **Caso de error**: Si el archivo seleccionado no existe (p. ej., se elimina mientras se visualiza la lista), se mostrará un mensaje de error indicando que no se encontró el archivo.

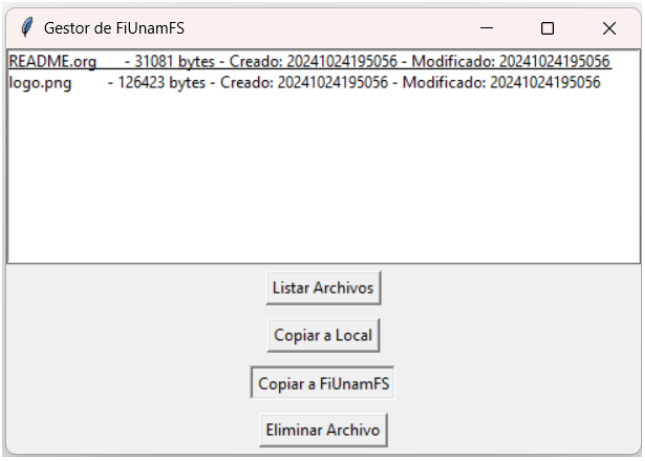


Mantengo esta ventana abierta para poder eliminar el archivo y mostrar el mensaje de error. Para ver como eliminar archivos ir a su apartado más abajo

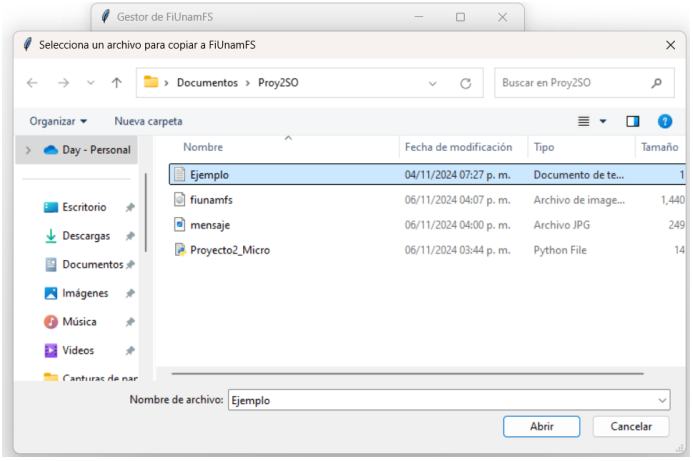


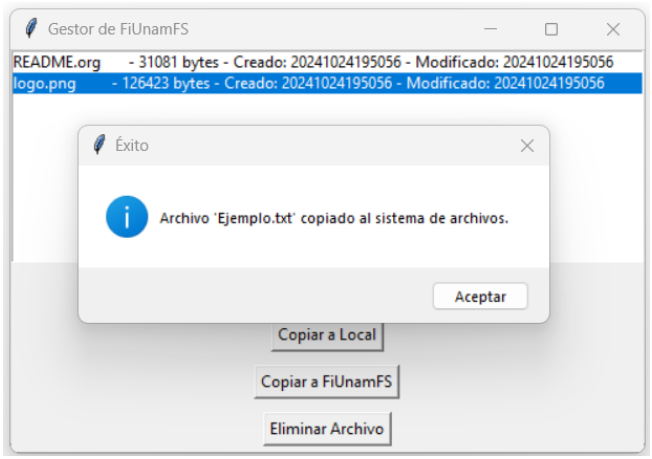
- **Caso de cancelación**: Si el usuario cancela el cuadro de diálogo de guardado, no se realizará ninguna acción y no se mostrará ningún mensaje.

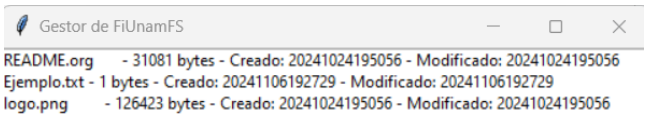
- **Copiar un Archivo desde el Sistema Local a FiUnamFS**
  - **Acción**: Presiona el botón "Copiar a FiUnamFS" y selecciona un archivo desde tu sistema local.



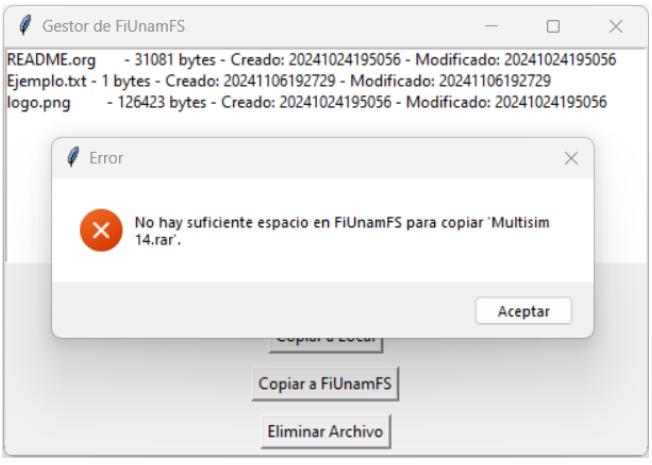
  - **Resultado esperado**: El archivo se copiará al sistema de archivos FiUnamFS y aparecerá un mensaje de éxito en la interfaz.



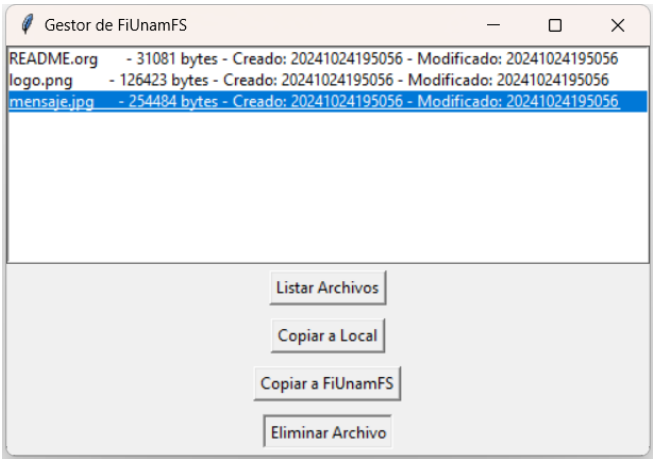


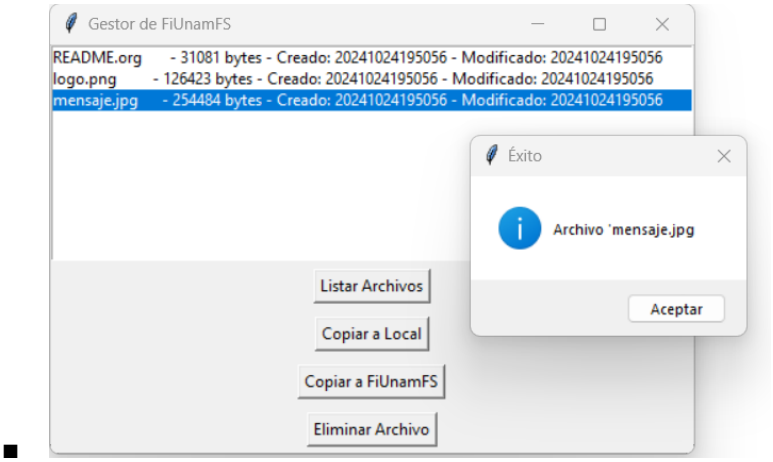


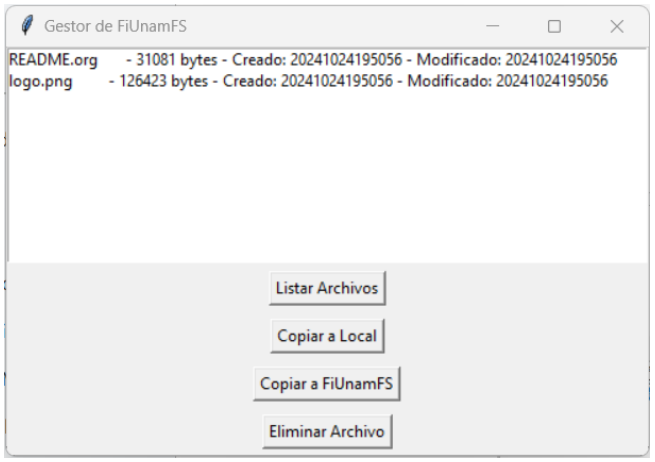
- **Caso de error**:
  - Si no hay suficiente espacio en FiUnamFS para el archivo, se mostrará un mensaje de error indicando que no hay espacio disponible.



  - Si no se encuentra un espacio libre en el directorio para almacenar la entrada del archivo, se mostrará un mensaje de error indicando que el directorio está lleno.
- **Caso de cancelación**: Si el usuario cancela la selección del archivo, no se realizará ninguna acción.
-

- **Eliminar un Archivo de FiUnamFS**
  - **Acción**: Selecciona un archivo de la lista y presiona el botón "Eliminar Archivo".
    - 

  - **Resultado esperado**: El archivo se marcará como eliminado y se mostrará un mensaje de éxito.
    - 

    - 

  - **Caso de error**: Si el archivo seleccionado no se encuentra en FiUnamFS (p. ej., si fue eliminado por otra operación), se mostrará un mensaje de error indicando que el archivo no fue encontrado.
  - **Caso de cancelación**: No aplicable, ya que el proceso de eliminación es inmediato tras la selección y confirmación.

Errores y Mensajes de Estado

- **Error de archivo inexistente**: Si el archivo fiunamfs.img no está presente en el directorio de trabajo, se mostrará un mensaje de error y se cancelarán todas las operaciones.
- **Archivo dañado o formato incorrecto**: Si el archivo fiunamfs.img tiene un formato que no coincide con la estructura esperada de FiUnamFS, se mostrará un mensaje de error y se detendrán las operaciones.

8. Instrucciones de Instalación y Ejecución

1. Descarga el archivo fiunamfs.img y colócalo en el mismo directorio que el código.
2. Asegúrate de tener Python 3.6+ instalado.
3. Ejecuta el programa con: python Proyecto2_Micro.py

9.Conclusiones

González Iniestra Emilio

El desarrollo del proyecto me permitió profundizar en la programación multihilo y la sincronización de procesos, aspectos fundamentales para garantizar la integridad de los datos y la eficiencia en sistemas concurrentes. Implementar hilos para realizar operaciones de E/S y utilizar threading.Lock fue esencial para evitar bloqueos y errores de acceso simultáneo a los archivos, lo cual fue un desafío que me hizo comprender la importancia de la planificación y la gestión adecuada de los recursos en sistemas operativos.

Asimismo, trabajar en la construcción de una interfaz gráfica de usuario con tkinter me ayudó a entender mejor cómo las herramientas de interfaz deben manejar múltiples operaciones en paralelo sin sacrificar la experiencia del usuario. Fue una oportunidad de combinar conceptos de diseño y programación para lograr un entorno interactivo que permite al usuario manejar el sistema de archivos de forma intuitiva y eficaz, consolidando mi habilidad para desarrollar aplicaciones de escritorio en Python.

El proceso de implementación y pruebas me hizo valorar la importancia de documentar y comentar el código detalladamente, especialmente en proyectos colaborativos. La claridad en el código y la estructura bien definida son clave para mantener la legibilidad y facilitar futuras actualizaciones. Este proyecto no solo reforzó mis conocimientos técnicos, sino que también me ayudó a desarrollar mejores prácticas de codificación y colaboración en equipo.

Suárez Guzmán Dayna Yarelly

La experiencia de trabajar en el proyecto me proporcionó una visión más amplia sobre el manejo y la creación de sistemas de archivos personalizados, enfocándome en aspectos prácticos y conceptuales del diseño de software. La utilización de hilos para la ejecución de tareas concurrentes mejoró mi comprensión sobre la sincronización y la protección de secciones críticas mediante threading.Lock, un conocimiento que considero invaluable para desarrollos futuros en entornos de programación más complejos.

Además, participar en el desarrollo de la interfaz gráfica con tkinter me permitió explorar cómo el diseño y la funcionalidad pueden integrarse para crear aplicaciones más interactivas y fáciles de usar. Aprender a gestionar la interacción del usuario con operaciones en segundo plano, sin que la aplicación se bloquee o se vuelva inoperante, fue uno de los aspectos más desafiantes y enriquecedores de este proyecto.

La colaboración y la comunicación dentro del equipo fueron elementos vitales para el éxito del proyecto. Este proceso me enseñó a planificar y dividir el trabajo de manera eficiente, asegurando que cada parte del proyecto se completara a tiempo y con la calidad requerida. Las lecciones aprendidas sobre el trabajo en equipo, la documentación minuciosa y el uso de buenas prácticas de programación fortalecieron mis conocimientos.