UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMAS OPERATIVOS

PROYECTO: (MICRO) SISTEMA DE ARCHIVOS MULTIHILOS

Alumnos:

- **De La Merced Soriano Uriel Benjamín**
- **Hernández Gutiérrez Carlos Mario**

PROFESOR: Ing. Gunnar Eyal Wolf Iszaevich

GRUPO: 6

FECHA DE ENTREGA: 19/03/2024

Descripción del Proyecto

Este proyecto consiste en desarrollar un micro-sistema de archivos multihilos para la Facultad de Ingeniería de la UNAM (FiUnamFS). Este sistema permitirá realizar operaciones básicas sobre un sistema de archivos simulado que se almacena en un archivo de longitud fija de 1440 Kilobytes, simulando un diskette tradicional.

El sistema de archivos debe cumplir con las siguientes especificaciones:

- Listar los contenidos del directorio.
- Copiar un archivo desde el FiUnamFS hacia nuestra computadora.
- Copiar un archivo desde nuestra computadora hacia el FiUnamFS.
- Eliminar un archivo del FiUnamFS.

Estructura del Proyecto

Nuestro proyecto está desarrollado en Python, ya que su sintaxis es sencilla y permite una mayor flexibilidad al crear y gestionar hilos de ejecución. Python también cuenta con bibliotecas robustas para manejar archivos y sincronización de hilos, lo que facilita la implementación de un sistema de archivos multihilos.

Descripción del Sistema de Archivos

Se nos proporcionó un archivo llamado fiunamfs.img, que simula un sistema de archivos y contiene tres archivos dentro. Nuestro objetivo es desarrollar un programa que interactúe con este sistema de archivos, realizando las funcionalidades anteriormente habladas. Si nuestro programa funciona correctamente, podremos visualizar y manipular los contenidos de "*fiunamfs.img*".

El proyecto estará estructurado de la siguiente manera:

Archivo principal que inicializa y una interfaz a nuestro sistema de archivos y se le dan las siguientes opciones a nuestro usuario para que elija que quiere realizar:

1. Listar contenidos de FIUNAMFS.
2. Copiar archivo desde FIUNAMFS al equipo local.
3. Copiar archivo desde el equipo local a FIUNAMFS.
4. Eliminar un archivo de FIUNAMFS.
5. Revisar información del disco.
6. Salir del programa.

Interfaz del usuario

Se definió un diccionario llamado "opciones_menu" donde cada tecla de función (F1-F6) está asociada a una tupla que contiene una descripción de la acción y la función correspondiente que se ejecutará cuando se seleccione esa opción.

Se define la función “capturar_tecla” que se ejecuta cada vez que se presiona una tecla. Verifica si la tecla presionada corresponde a una de las opciones del menú y ejecuta la función asociada.

```python
# Definir las opciones del menú
opciones_menu = {
    "F1": ("F1. Listar contenido de FiUnamFS", listar_contenido_fiunamfs),
    "F2": ("F2. Copiar archivo desde FiUnamFS a tu equipo", copiar_archivo_fiunamfs),
    "F3": ("F3. Copiar archivo desde tu equipo a FiUnamFS", copiar_archivo_equipo_fiunamfs),
    "F4": ("F4. Eliminar archivo de FiUnamFS", eliminar_archivo_fiunamfs),
    "F5": ("F5. Revisar información del disco", revisar_informacion_disco),
    "F6": ("F6. Salir del programa", salir_programa)
}

def capturar_tecla(event):
    key = event.keysym
    if key in opciones_menu:
        opciones_menu[key][1]()

def mostrar_menu_principal():
    # Crear la etiqueta de instrucciones
    label = tk.Label(root, text="Por favor, selecciona una opción del menú (F1-F6):", fg="white", bg="black", font=("Arial", 14))
    label.pack(pady=20)  # Agregar un espacio en la parte superior e inferior

    # Crear los botones para cada opción del menú
    for key, (text, command) in opciones_menu.items():
        button = tk.Button(root, text=text, command=command, bg="black", fg="white", font=("Arial", 12))
        button.pack(pady=10)  # Agregar un espacio entre los botones

    # Capturar la entrada del usuario
    root.bind("<Key>", capturar_tecla)

# Crear la ventana principal
root = tk.Tk()
root.title("Proyecto: (Micro) sistema de archivos multihilos")
root.geometry("600x400")  # Establecer el tamaño de la ventana
root.configure(bg="black")  # Establecer el fondo negro


# Función para mostrar el menú principal cuando se inicia el programa
mostrar_menu_principal()

# Ejecutar el bucle de eventos
root.mainloop()
```

Código interfaz de usuario

Constantes iniciales en el código

- **TAMANIO_SECTOR**: Representa el tamaño de un sector en bytes. En este caso, es de 512 bytes.
- **TAMANIO_CLUSTER**: Indica el tamaño de un clúster, calculado multiplicando el tamaño del sector por 4. Esto se debe a que cada clúster consta de 4 sectores consecutivos.
- **TAMANIO_ENTRADA_DIRECTORIO**: Es el tamaño de una entrada en el directorio del sistema de archivos *FiUNAMFS*, establecido en 64 bytes.
- **NUMERO_ENTRADAS_DIRECTORIO**: Representa el número total de entradas en el directorio del sistema de archivos. Se calcula multiplicando el número de clusters (4) por el número de entradas por cluster (16).
- **DIRECTORIO_INICIO:** Indica la posición inicial del directorio en el archivo de imagen del sistema de archivos, que se encuentra en el primer clúster.
- **DIRECTORIO_FIN**: Es la posición final del directorio en el archivo de imagen del sistema de archivos, que está ubicado en el cuarto clúster.

- **Semáforo**: Se utiliza un semáforo para controlar el acceso concurrente al recurso compartido. En este caso, se inicializa con un valor de 1, lo que significa que solo un hilo puede acceder al recurso compartido a la vez.

```
# Constantes
TAMANIO_SECTOR = 512
TAMANIO_CLUSTER = TAMANIO_SECTOR * 4
TAMANIO_ENTRADA_DIRECTORIO = 64
NUMERO_ENTRADAS_DIRECTORIO = 16  # 4 clusters * 16 entradas por cluster
DIRECTORIO_INICIO = 1 * TAMANIO_CLUSTER
DIRECTORIO_FIN = 4 * TAMANIO_CLUSTER

# Semáforo para controlar el acceso concurrente al recurso compartido
semaforo = threading.Semaphore(value=1)
```

Constantes del código

- **Función leer_entero**:Esta función lee un entero de 32 bits desde el archivo fuente en una posición dada. Utiliza la función int.from_bytes() para convertir los bytes leídos en un entero en formato little-endian.
- **Función leer_cadena**: Lee una cadena de texto ASCII de longitud especificada desde el archivo fuente en una posición dada. Los bytes leídos se decodifican utilizando el formato ASCII.
- Función leer_cadena2: Similar a *leer_cadena*, pero permite especificar un archivo de origen diferente.

```
# Función para leer enteros desde el archivo fuente
def leer_entero(posicion, longitud):
    with open("fiunamfs.img", "rb") as archivo:
        archivo.seek(posicion)
        bytes_entero = archivo.read(longitud)
        entero = int.from_bytes(bytes_entero, byteorder='little')
        return entero

# Función para leer cadenas desde el archivo fuente
def leer_cadena(posicion, longitud):
    with open("fiunamfs.img", "rb") as archivo:
        archivo.seek(posicion)
        bytes_cadena = archivo.read(longitud)
        cadena = bytes_cadena.decode("ascii")
        return cadena

# Función para leer una cadena de longitud fija desde el archivo
def leer_cadena2(archivo, posicion, longitud):
    archivo.seek(posicion)
    cadena = archivo.read(longitud)
    return cadena.decode('ascii').rstrip('\x00')
```

Funciones para leer enteros y cadenas

Función Iniciar:

Esta función inicia el sistema de archivos FiUNAMFS. Lee información importante del sistema, como el nombre del sistema, la versión, la etiqueta del volumen, el tamaño del

clúster y otros detalles relevantes. Luego, verifica si la versión y el nombre del sistema son correctos antes de imprimir la información del sistema.

```python
# Función para validar y mostrar información del servidor
def iniciar():
    nombre_sistema = leer_cadena(0, 9)
    version = leer_cadena(10, 5)
    etiqueta_volumen = leer_cadena(20, 20)
    tamanio_cluster = leer_entero(40, 4)
    num_clusters_directorio = leer_entero(45, 4)
    num_total_clusters = leer_entero(50, 4)

    if version != '24-2':
        print("La versión no es la adecuada, el sistema sólo admite 24-2")
        return False
    if nombre_sistema != 'FiUnamFS':
        print("El nombre del sistema no es el adecuado, el sistema sólo admite FiUnamFS")
        return False
    # Imprimir las características del sistema
    print("=== Información del Sistema de Archivos ===")
    print(f"Nombre del sistema: {nombre_sistema}")
    print(f"Versión: {version}")
    print(f"Etiqueta del volumen: {etiqueta_volumen}")
    print(f"Tamaño del cluster: {tamanio_cluster} bytes")
    print(f"Número de clusters en el directorio: {num_clusters_directorio}")
    print(f"Número total de clusters: {num_total_clusters}")
```

Función iniciar

Listar contenidos de FIUNAMFS

1. **Apertura del Archivo**:Se abre el archivo *fiunamfs.img* en modo de lectura binaria (rb). Este archivo simula el sistema de archivos FiUnamFS.

2. **Lectura del Directorio**: El directorio está ubicado en los clusters 1 a 4. Se itera sobre estos clusters para leer el contenido del directorio '*inicio_cluster'* calcula la posición inicial de cada cluster en el archivo.

3. **Lectura de Entradas del Directorio**: Dentro de cada cluster, se itera sobre las entradas del directorio "*entrada_posicion*" calcula la posición de cada entrada en el archivo.

4. **Lectura del Tipo de Archivo**: Se lee el tipo de archivo en la posición de la entrada. Si el tipo de archivo es '-', indica que es un archivo regular.

5. **Lectura de los Detalles del Archivo**:
    1. *Nombre del Archivo*: Se lee desde la posición *entrada_posicion* + 1 y tiene una longitud de 15 bytes.
    2. *Tamaño del Archivo*: Se lee a continuación del nombre y se desempaqueta utilizando *struct.unpac*k con formato "<I" (entero de 32 bits en little-endian).
    3. *Cluster Inicial*: Se lee a continuación del tamaño del archivo y también se desempaqueta con *struct.unpack*.

4. *Hora de Creación*: Se lee desde la posición entrada_posicion + 24 y tiene una longitud de 14 bytes.
5. *Hora de Modificación*: Se lee desde la posición entrada_posicion + 38 y también tiene una longitud de 14 bytes.

6. Finalmente, se imprime la información del archivo: nombre, tamaño en bytes, cluster inicial, hora de creación y hora de última modificación.

```python
# Función para listar los contenidos del directorio
def listar_contenidos_directorio():
    with open("fiunamfs.img", "rb") as archivo:
        # Leer el directorio ubicado en los clusters 1 a 4
        for cluster in range(1, 5):
            inicio_cluster = (cluster - 1) * TAMANIO_CLUSTER
            for i in range(NUMERO_ENTRADAS_DIRECTORIO):
                entrada_posicion = inicio_cluster + i * TAMANIO_ENTRADA_DIRECTORIO
                tipo_archivo = leer_cadena2(archivo, entrada_posicion, 1)
                if tipo_archivo == '-':  # Archivo regular
                    nombre_archivo = leer_cadena2(archivo, entrada_posicion + 1, 15)
                    tamano_archivo = struct.unpack("<I", archivo.read(4))[0]
                    cluster_inicial = struct.unpack("<I", archivo.read(4))[0]
                    hora_creacion = leer_cadena2(archivo, entrada_posicion + 24, 14)
                    hora_modificacion = leer_cadena2(archivo, entrada_posicion + 38, 14)
                    print(f"Nombre: {nombre_archivo}, Tamaño: {tamano_archivo} bytes, Cluster inicial: {cluster_inicial}, Creado: {hora_creacion}, Modificado:
```

Código para listar contenido

Copiar archivo desde FIUNAMFS al equipo local

Esta función se encarga de encontrar y leer un archivo desde el sistema de archivos *FiUnamFS* y devolver su contenido.

Funcion **copiar_archivo_desde_fiunamfs**: Esta función se encarga de encontrar y leer un archivo desde el sistema de archivos FiUnamFS y devolver su contenido.

1. **Buscar el Archivo en el Directorio:**El directorio está distribuido en los clusters 1 a 4. Se itera sobre estos clusters y se calcula la posición inicial de cada cluster con *inicio_cluster*.
2. **Leer Entradas del Directorio**:Dentro de cada cluster, se itera sobre las entradas del directorio. "*entrada_posicion*" calcula la posición de cada entrada, y *tipo_archivo* lee el tipo de archivo (regular o vacío).
3. **Verificar el Nombre del Archivo**:Se verifica si el archivo es regular o está vacío y se compara el nombre del archivo actual con el nombre del archivo que se desea copiar.
4. **Leer y Devolver el Contenido del Archivo**:Si se encuentra el archivo, se leen su tamaño y su cluster inicial. Luego, se posiciona en el cluster inicial y se lee el contenido del archivo, que se devuelve como resultado.

```python
# Función para copiar un archivo desde FiUnamFS a tu sistema
def copiar_archivo_desde_fiunamfs(nombre_archivo):
    with open("fiunamfs.img", "rb") as archivo:
        # Buscar el archivo en el directorio
        for cluster in range(1, 5):
            inicio_cluster = (cluster - 1) * TAMANIO_CLUSTER
            for i in range(NUMERO_ENTRADAS_DIRECTORIO):
                entrada_posicion = inicio_cluster + i * TAMANIO_ENTRADA_DIRECTORIO
                tipo_archivo = leer_cadena2(archivo, entrada_posicion, 1)
                if tipo_archivo == '-' or tipo_archivo == '/':  # Archivo regular o vacío
                    nombre_archivo_actual = leer_cadena2(archivo, entrada_posicion + 1, 15)
                    if nombre_archivo_actual.strip() == nombre_archivo:
                        tamano_archivo = struct.unpack("<I", archivo.read(4))[0]
                        cluster_inicial = struct.unpack("<I", archivo.read(4))[0]
                        archivo.seek(TAMANIO_CLUSTER * cluster_inicial)
                        contenido_archivo = archivo.read(tamano_archivo)
                        return contenido_archivo
    return None
```

Función copiar archivo desde FiUnamFS al sistema

Función copiar_archivo_thread

Esta función maneja la copia del archivo desde *FiUnamFS* al sistema local en un hilo separado, utilizando un semáforo para asegurar la sincronización.

1. **Adquirir el Semáforo**: Se adquiere un semáforo para asegurar que solo un hilo acceda al recurso compartido a la vez.
2. **Copiar el Archivo**: Se llama a la función "*copiar_archivo_desde_fiunamfs*" para obtener el contenido del archivo.
3. **Seleccionar el Directorio de Destino**: Si el archivo fue encontrado, se abre un diálogo para que el usuario seleccione el directorio donde desea guardar el archivo.
4. **Guardar el Archivo**: Si se selecciona un directorio, se guarda el archivo en la ubicación especificada. Si no se selecciona ningún directorio, se muestra un mensaje de error.
5. **Liberar el Semáforo**: Finalmente, se libera el semáforo, permitiendo que otros hilos puedan acceder al recurso compartido.

```python
# Función para manejar la copia del archivo en un hilo separado
def copiar_archivo_thread(nombre_archivo):
    semaforo.acquire()
    try:
        contenido_archivo = copiar_archivo_desde_fiunamfs(nombre_archivo)
        if contenido_archivo:
            # Abrir un diálogo para que el usuario elija el directorio de destino
            root = tk.Tk()
            root.withdraw()  # Ocultar la ventana principal
            directorio_destino = filedialog.askdirectory(title="Seleccionar directorio de destino")
            if directorio_destino:
                # Guardar el archivo en el directorio seleccionado
                ruta_archivo_destino = os.path.join(directorio_destino, nombre_archivo)
                with open(ruta_archivo_destino, "wb") as archivo_destino:
                    archivo_destino.write(contenido_archivo)
                print(f"Archivo '{nombre_archivo}' copiado exitosamente a '{ruta_archivo_destino}'.")
            else:
                print("No se seleccionó ningún directorio.")
        else:
            print(f"Archivo '{nombre_archivo}' no encontrado en FiUnamFS.")
    finally:
        semaforo.release()
```

Función copiar archivo

Copiar archivo desde el equipo local a FIUNAMFS

Función encontrar_espacio_libre

Esta función busca espacio libre en el disco donde se pueda almacenar el nuevo archivo. Busca espacio libre de tamaño en el archivo del sistema de archivos **fiunamfs.img**.

- Se posiciona en el inicio del espacio de datos y busca una secuencia de bytes nulos (b'\x00') del tamaño necesario.
- Si encuentra un espacio libre, devuelve la posición. Si no, lanza un error.

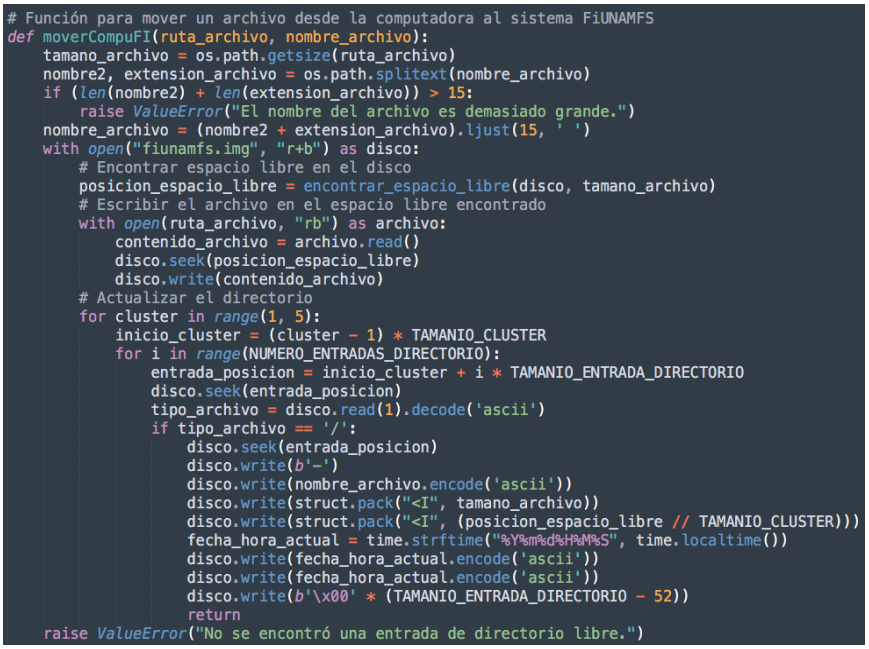
```python
# Función para encontrar espacio libre en el disco
def encontrar_espacio_libre(disco, tamano_archivo):
    disco.seek(DIRECTORIO_FIN)  # Inicio del espacio de datos
    espacio_libre = b'\x00' * tamano_archivo  # Espacio libre requerido
    datos = disco.read()  # Leer todos los datos
    indice = datos.find(espacio_libre)  # Buscar la primera aparición del espacio libre
    if indice != -1:
        return DIRECTORIO_FIN + indice
    else:
        raise ValueError("No se encontró espacio suficiente para el archivo.")
```

Función encontrar espacio

Función moverCompuFI

Esta función maneja el proceso de mover un archivo desde la computadora al sistema FiUnamFS.

1. **Validar el Nombre del Archivo**: Se verifica que el nombre del archivo (nombre + extensión) no exceda los 15 caracteres permitidos. Si lo excede, lanza un error.
2. **Buscar Espacio Libre**:Abre el archivo *fiunamfs.img* y utiliza *encontrar_espacio_libre* para encontrar una posición libre en el disco.
3. **Escribir el Archivo en el Espacio Libre**:Lee el contenido del archivo local y lo escribe en la posición libre encontrada en el disco de FiUnamFS.
4. **Actualizar el Directorio**:Recorre las entradas del directorio (clusters 1 a 4) para encontrar una entrada vacía ('/').Actualiza la entrada del directorio con la información del nuevo archivo (tipo de archivo, nombre, tamaño, cluster inicial, fecha y hora de creación y modificación).



```python
# Función para mover un archivo desde la computadora al sistema FiUNAMFS
def moverCompuFI(ruta_archivo, nombre_archivo):
    tamano_archivo = os.path.getsize(ruta_archivo)
    nombre2, extension_archivo = os.path.splitext(nombre_archivo)
    if (len(nombre2) + len(extension_archivo)) > 15:
        raise ValueError("El nombre del archivo es demasiado grande.")
    nombre_archivo = (nombre2 + extension_archivo).ljust(15, ' ')
    with open("fiunamfs.img", "r+b") as disco:
        # Encontrar espacio libre en el disco
        posicion_espacio_libre = encontrar_espacio_libre(disco, tamano_archivo)
        # Escribir el archivo en el espacio libre encontrado
        with open(ruta_archivo, "rb") as archivo:
            contenido_archivo = archivo.read()
            disco.seek(posicion_espacio_libre)
            disco.write(contenido_archivo)
        # Actualizar el directorio
        for cluster in range(1, 5):
            inicio_cluster = (cluster - 1) * TAMANIO_CLUSTER
            for i in range(NUMERO_ENTRADAS_DIRECTORIO):
                entrada_posicion = inicio_cluster + i * TAMANIO_ENTRADA_DIRECTORIO
                disco.seek(entrada_posicion)
                tipo_archivo = disco.read(1).decode('ascii')
                if tipo_archivo == '/':
                    disco.seek(entrada_posicion)
                    disco.write(b'-')
                    disco.write(nombre_archivo.encode('ascii'))
                    disco.write(struct.pack("<I", tamano_archivo))
                    disco.write(struct.pack("<I", (posicion_espacio_libre // TAMANIO_CLUSTER)))
                    fecha_hora_actual = time.strftime("%Y%m%d%H%M%S", time.localtime())
                    disco.write(fecha_hora_actual.encode('ascii'))
                    disco.write(fecha_hora_actual.encode('ascii'))
                    disco.write(b'\x00' * (TAMANIO_ENTRADA_DIRECTORIO - 52))
                    return
    raise ValueError("No se encontró una entrada de directorio libre.")
```

Función mover archivo a FIUNAMFS

Función copiar_archivo_con_semaforo

Esta función maneja la copia del archivo utilizando threading y semáforos para asegurar la sincronización

1. **Crear un semáforo:** Se crea un semáforo para controlar el acceso al recurso compartido.
2. **Definir la Tarea:**
    - Dentro de la tarea, se adquiere el semáforo.
    - Se abre un diálogo con Tkinter para que el usuario seleccione un archivo local.
    - Si se selecciona un archivo, se llama a moverCompuFI para copiar el archivo al sistema FiUnamFS.
    - Finalmente, se libera el semáforo.
3. **Crear y Ejecutar el Hilo**:
    - Se crea un hilo que ejecuta la tarea definida.
    - Se inicia el hilo y se espera a que termine con join.

```python
# Función para manejar el archivo con threading y semáforo
def copiar_archivo_con_semaforo():
    semaforo = threading.Semaphore(value=1)

    def tarea():
        semaforo.acquire()
        try:
            # Usar tkinter para seleccionar el archivo
            root = tk.Tk()
            root.withdraw()  # Ocultar la ventana principal de tkinter
            ruta_archivo = filedialog.askopenfilename(title="Selecciona un archivo")
            if not ruta_archivo:
                print("No se seleccionó ningún archivo.")
                return
            nombre_archivo = os.path.basename(ruta_archivo)
            moverCompuFI(ruta_archivo, nombre_archivo)
        finally:
            semaforo.release()

    hilo = threading.Thread(target=tarea)
    hilo.start()
    hilo.join()
```

Función copiar archivo

Eliminar un archivo de FIUNAMFS

Esta función busca y elimina un archivo específico del sistema de archivos FiUNAMFS.

1. **Preparación del Nombre del Archivo**:El nombre del archivo se ajusta a una longitud fija de 15 caracteres..
2. **Apertura del Archivo del Sistema**:Se abre el archivo *fiunamfs.img* en modo lectura y escritura binaria (r+b).
3. **Búsqueda del Archivo en el Directorio:**
    a. Itera sobre los clusters del sistema FiUNAMFS que contienen el directorio, buscando el archivo a eliminar.
    b. Se calcula la posición inicial de cada cluster en el archivo.
4. **Iteración sobre las Entradas del Directorio:**
    a. Dentro de cada cluster, se itera sobre las entradas del directorio.
    b. Se calcula la posición de cada entrada en el archivo.
5. **Lectura del Tipo de Archivo**: Se lee el tipo de archivo en la posición de la entrada. Si el tipo de archivo es '-', indica que es un archivo regular.
6. **Comparación del Nombre del Archivo:** Se compara el nombre del archivo en la entrada actual con el nombre del archivo especificado para borrar.
7. **Marcado del Archivo como Vacío:** Si se encuentra el archivo, se marca la entrada del directorio como vacía escribiendo '/' en la posición correspondiente y rellenando el resto de la entrada con caracteres '#' para indicar que está disponible para su uso futuro.
8. **Manejo de Casos de Error** :Si el archivo no se encuentra en el sistema FiUNAMFS, se imprime un mensaje de error indicando que el archivo no fue encontrado.

```python
# Función para borrar un archivo del sistema FiUNAMFS
def borrar_archivo(nombre_archivo):
    nombre_archivo = nombre_archivo.ljust(15, ' ')
    with open("fiunamfs.img", "r+b") as disco:
        # Buscar el archivo en el directorio
        for cluster in range(1, 5):
            inicio_cluster = (cluster - 1) * TAMANIO_CLUSTER
            for i in range(NUMERO_ENTRADAS_DIRECTORIO):
                entrada_posicion = inicio_cluster + i * TAMANIO_ENTRADA_DIRECTORIO
                disco.seek(entrada_posicion)
                tipo_archivo = disco.read(1).decode('ascii')
                if tipo_archivo == '-':
                    archivo_nombre = leer_cadena(disco, entrada_posicion + 1, 15)
                    if archivo_nombre == nombre_archivo:
                        # Marcar la entrada del archivo como vacía
                        disco.seek(entrada_posicion)
                        disco.write(b'/')  # Marcar como vacío
                        disco.write(b' ' * 63)  # Rellenar el resto de la entrada con '#'
                        return
    print("Archivo no encontrado en el sistema FiUNAMFS.")
```

Función eliminar archivo

Por términos de “código limpio”, separamos las funciones principales de la interfaz gráfica, ya que el sistema sigue siendo el mismo, sólo cambia el método de entrada de los datos en algunos casos, y claro se inicia una ventana adicional para todas las opciones disponibles.

Información del disco

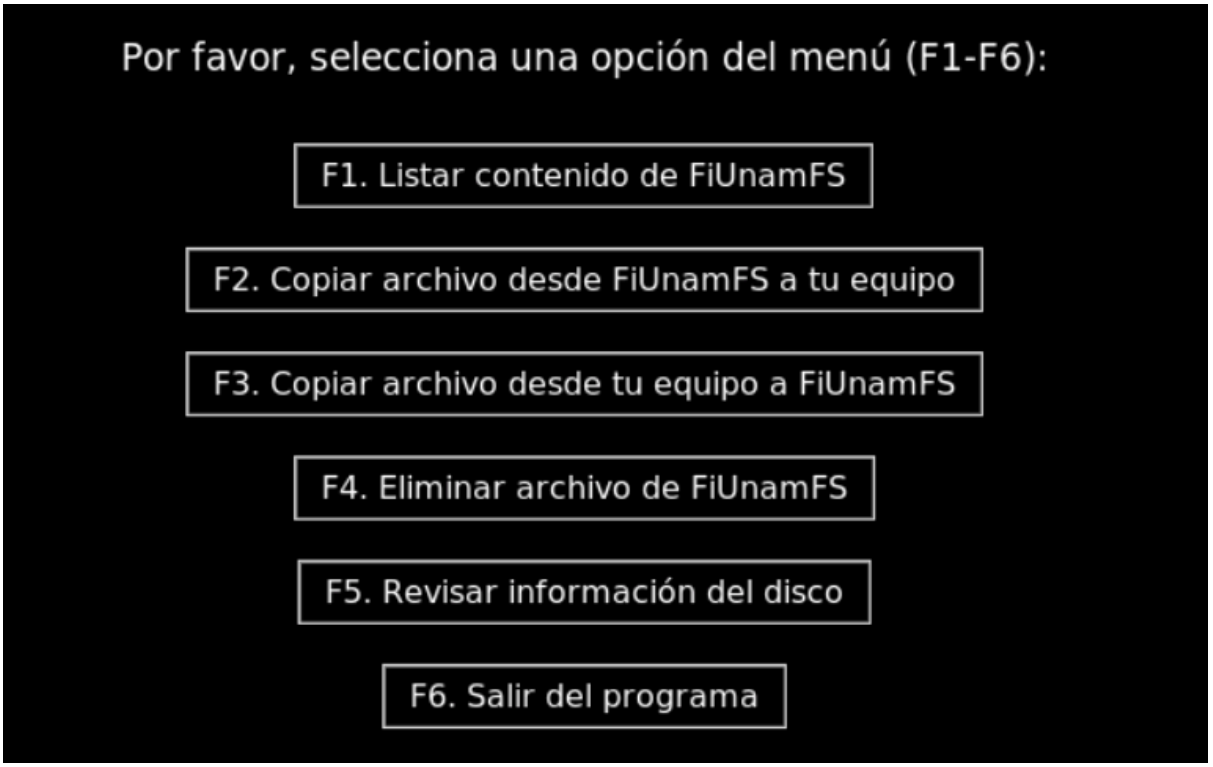
Abre una nueva ventana que muestra la información del sistema de archivos. La ventana incluye un encabezado y la información obtenida de la función iniciar(). El usuario puede regresar al menú principal presionando F1.

```python
# Función para revisar información del disco
def revisar_info_disco():
    def regresar_menu(event=None):
        ventana_informacion.destroy()  # Destruir la ventana actual
        root.deiconify()  # Mostrar la ventana principal

    root.withdraw()  # Ocultar la ventana principal

    ventana_informacion = tk.Toplevel(root)
    ventana_informacion.title("Información del Sistema de Archivos")
    ventana_informacion.geometry("800x600")
    ventana_informacion.configure(bg="black")

    encabezado = "Programa creado por De La Merced Soriano Uriel Benjamín y Hernández Gutiérrez Carlos Mario"
    informacion = iniciar()
    mensaje_completo = f"{encabezado}\n\n{informacion}\nF1 Regresar a inicio"

    etiqueta_informacion = tk.Label(ventana_informacion, text=mensaje_completo, font=("Arial", 12), bg="black", fg="white", justify=tk.LEFT)
    etiqueta_informacion.pack(pady=10, padx=10)

    ventana_informacion.bind("<F1>", regresar_menu)
"""
```

Función para mostrar la información del disco.

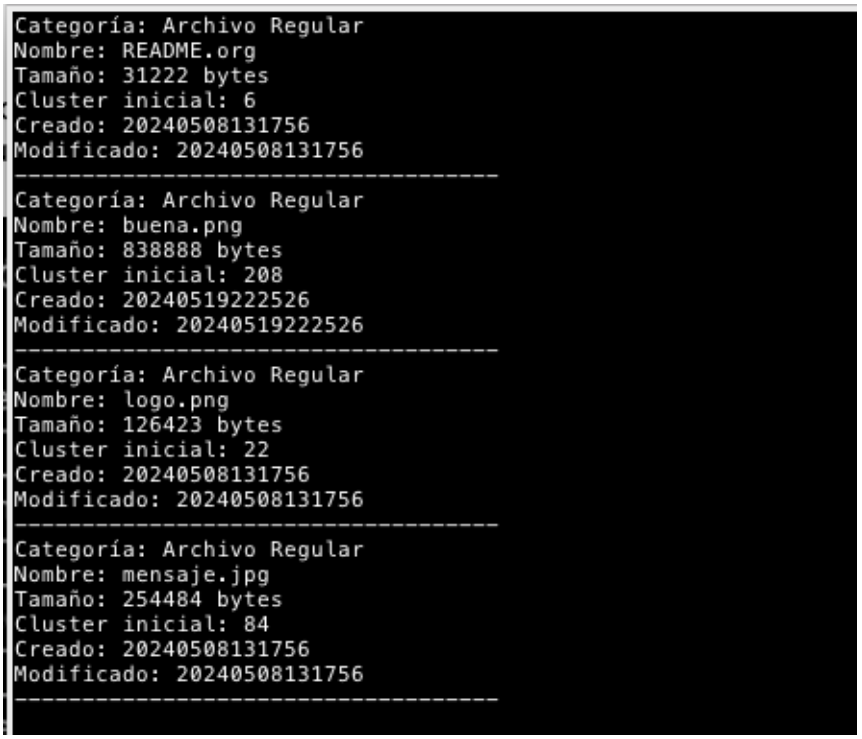
Programa en ejecución

Se muestra nuestro programa con las diferentes opciones a elegir
También se pueden elegir las opciones mediante pulsar las teclas F1 a F6 o dando click en el botón



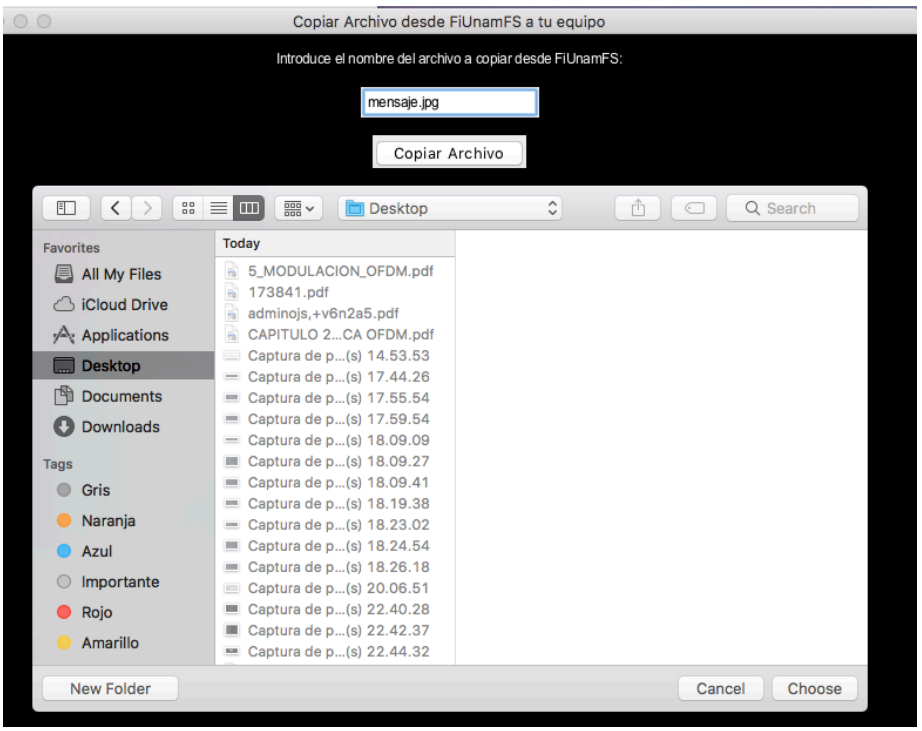
Interfaz inicial

Mostramos los archivos que se encuentran listados



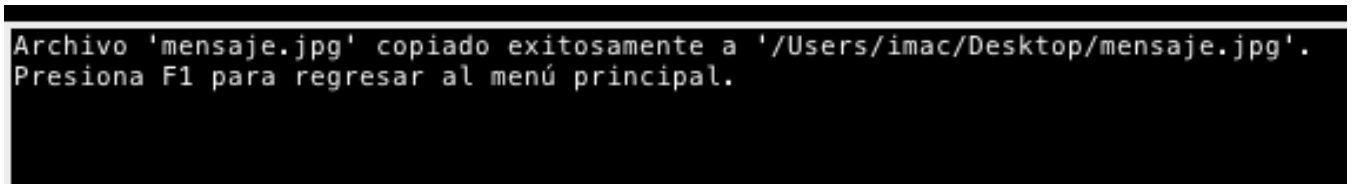
Lista de archivos de FIUnamFS

Este sistema oculta la ventana de inicio para abrir las opciones, y una vez le damos a regresar a inicio mediante la tecla “F1”, este destruye la ventana en la que estábamos y regresa a inicio. En los casos de la opción uno y cinco, se tiene que dar click con el mouse de nuestra computadora para que la tecla se registre en la ventana y la pueda cerrar, esto pasa porque como no está cerrada, sino oculta la ventana de inicio, el sistema seguirá en el sistema de inicio.
Cuando le damos a copiar archivo se nos abre una ventana para elegir en donde lo queremos guardar:



Copiar archivo a nuestro equipo

Una vez finalizado el proceso nos sale el mensaje que el archivo fue copiado exitosamente



Mensaje copiado exitosamente a la computadora

La imagen que se obtuvo fue la siguiente:

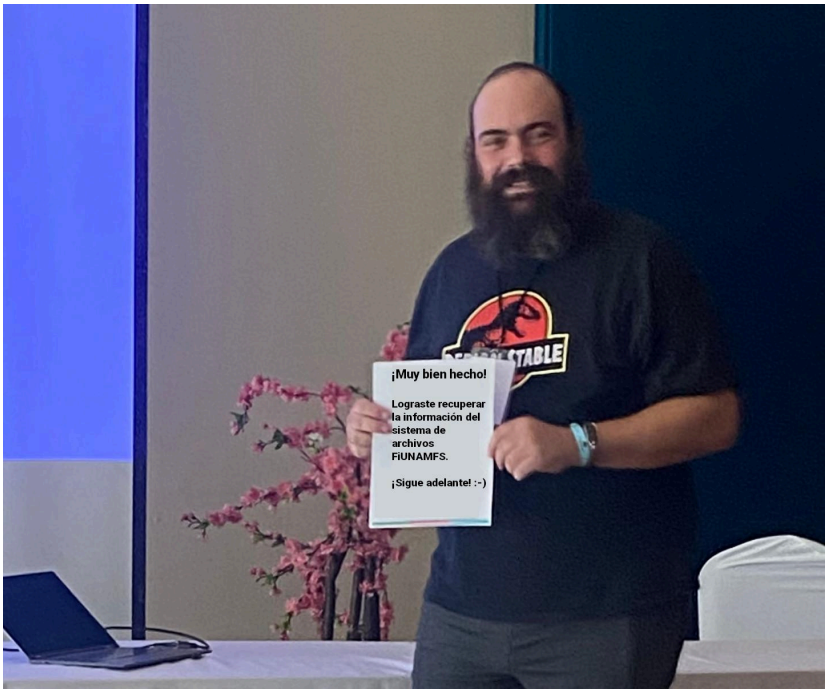
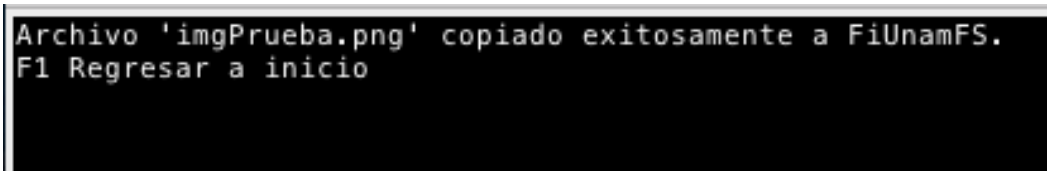


Imagen obtenida de FiunamFS

Copiamos un archivo de nuestra computadora a FiUnamFS. Se usará una imagen llamada “imgPrueba.png”, para subir imagenes al disco de archivos, el nombre de este deberá de ser menor a 15 caracteres, de lo contrario no se podrá pasar el archivo, en el otro caso, cuando es menor, el programa rellena los demás bits con espacios

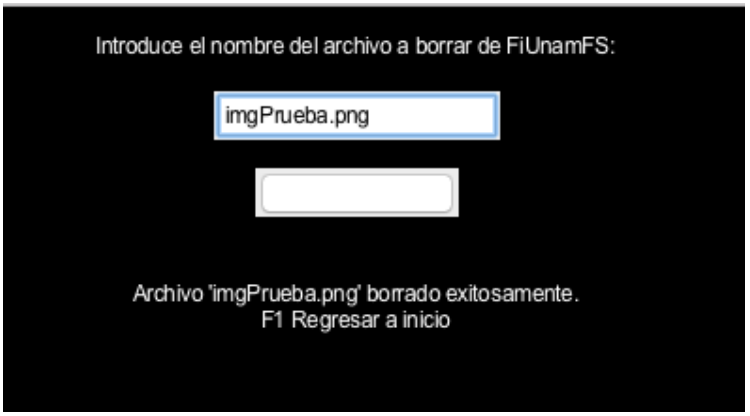


Mensaje copiado exitosamente a FiUnamFS

Revisamos que la imagen se encuentre en nuestra lista de contenido.

Lista de archivos con la nueva imagen

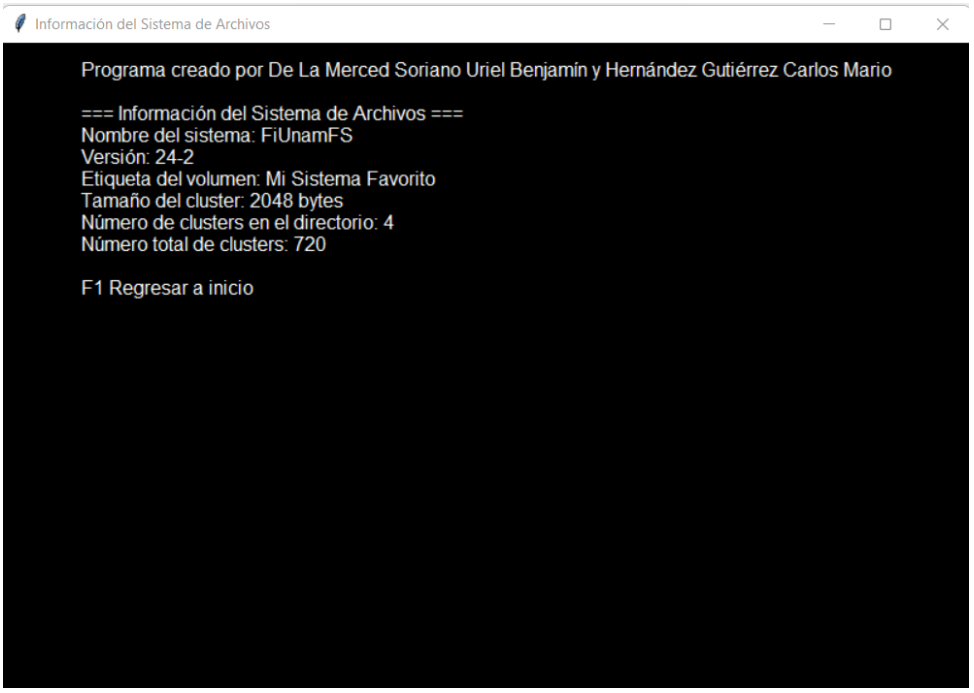
Ahora vamos a borrar el mismo archivo que subimos



Borrado de un archivo de FiUnamFS

Sólo podemos borrar los archivos que ingresamos desde nuestro dispositivo, ya que los archivos previamente guardados en el sistema no nos fué posible.

Por último revisamos la información de nuestro disco



Información del sistema de archivos

Con esto concluimos con la documentación de nuestro proyecto, abarcando los requerimientos del proyecto y realizando las pruebas necesarias para determinar el archivo final.