



UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO
FACULTAD DE INGENIERÍA



SISTEMAS OPERATIVOS

Profesor: Ing. Gunnar Eyal Wolf Iszaevich

Grupo: 05

Proyecto 2

“Microsistema de archivos”

Alumno:

Aquino Lozada Gabriela

Fecha de entrega:

07/Noviembre/2024

Semestre 2025-1

Introducción:

Este código implementa un sistema de archivos llamado **FiUnamFS**. El objetivo del programa es interactuar con una imagen binaria (fiunamfs.img) que representa el sistema de archivos. El programa contiene un menú para realizar operaciones básicas:

1. **Lectura del superbloque:** Verifica la integridad del sistema de archivos leyendo su nombre y versión.
2. **Listado de archivos:** Muestra los archivos que son almacenados en el directorio del sistema de archivos.
3. **Extracción de archivos:** Copia los archivos desde el sistema a la máquina local.
4. **Inserción de archivos:** Agrega archivos desde el sistema local al sistema de archivos.
5. **Eliminación de archivos:** Marca archivos en la imagen como eliminados.
6. **Desfragmentación:** Reorganiza los datos en la imagen para eliminar fragmentación y mejorar el ordenamiento de los archivos.

Fue desarrollado en Python versión 3.13, en la cual se hizo uso de los módulos predeterminados de os y struct, teniendo una compatibilidad en las versiones de Python.

Características del programa:

- Lenguaje de programación: Python, versión: 3.13
- Entorno de ejecución: Se puede ejecutar en cualquier SO siempre y cuando tenga instalado python.
- Entorno de Desarrollo: IDE; Visual Studio Code, SO; Ubuntu 22.04

Funcionamiento del programa:

La función read_superblock(image_name) verifica si el archivo es una imagen del sistema de archivos FiUnamFS. Lo que hace, es abrir el archivo de imagen especificado por image_name y comienza a leer los bytes para identificar el nombre y su versión. Si al comparar los datos que se determinaron que fueran los esperados "FiUnamFS" Y "25-1" coinciden, en automático se considera valida. Si este no fuera el caso y no llegaran a coincidir, arroja un mensaje de error, indicando que no se trata de un archivo valido de FiUnamFs.

```
# Función para leer el superbloque de la imagen de sistema de archivos.
def read_superblock(image_path):
    with open(image_path, 'rb') as img: # Abre la imagen en modo de lectura binaria.
        img.seek(0) # Posiciona el cursor al inicio del archivo.
        fs_name = img.read(8).decode('ascii').strip() # Lee y decodifica el nombre del sistema de archivos.
        img.seek(10) # Posiciona el cursor en la posición 10.
        fs_version = img.read(5).decode('ascii').rstrip('\x00').strip() # Lee y decodifica la versión del sistema.
        print(f"Sistema de archivos: {fs_name}, Versión: {fs_version}") # Muestra el nombre y versión del sistema.

    # Verifica si los valores leídos coinciden con los esperados.
    if fs_name != "FiUnamFS" or fs_version != "25-1":
        print("Valores incorrectos, se esperaba FiUnamFS y 25-1") # Mensaje si el nombre o versión no coinciden
    else:
        print("Superbloque válido") # Mensaje si el superbloque es válido.
```

La función `list_directory(image_name)` su función es listar los archivos que se encuentran en FiUnamFS. Al abrir el archivo, este recorre el directorio en búsqueda de los archivos, posteriormente imprimirá el nombre de los archivos existentes.

```
# Función para listar los archivos del directorio en la imagen del sistema.
def list_directory(image_path):
    with open(image_path, 'rb') as img: # Abre la imagen en modo de lectura binaria.
        img.seek(DIR_START) # Posiciona el cursor al inicio del directorio.
        for _ in range(DIR_SIZE // ENTRY_SIZE): # Itera sobre cada entrada del directorio.
            entry = img.read(ENTRY_SIZE) # Lee una entrada completa.
            file_type = entry[0:1] # Identifica el tipo de archivo (byte 0).
            name = entry[1:16].decode('ascii').rstrip() # Extrae y decodifica el nombre del archivo (bytes 1-15).
            if file_type == b'.' and name: # Verifica si la entrada representa un archivo válido.
                print(f"Archivo: {name}") # Imprime el nombre del archivo.
```

La función `extract_file(image_name, internal_filename, external_path)` va a extraer un archivo desde el sistema FiUnamFS hacia el sistema local. Con la ayuda de `internal_filename` va a buscar el nombre del archivo dentro del sistema, si este es encontrado con éxito, leerá el contenido para posterior guardarlos en un nuevo archivo en el sistema local, con la ruta especificada por el `external_path`. Si este no fue encontrado, en automático se imprimirá un mensaje de error.

```
# Función para extraer un archivo desde FiUnamFS al sistema local.
def extract_file(image_path, filename, destination):
    with open(image_path, 'rb') as img: # Abre la imagen en modo de lectura binaria.
        img.seek(DIR_START) # Posiciona el cursor al inicio del directorio.
        for _ in range(DIR_SIZE // ENTRY_SIZE): # Itera sobre cada entrada del directorio.
            entry = img.read(ENTRY_SIZE) # Lee una entrada completa.
            file_type = entry[0:1] # Tipo de archivo (byte 0).
            if file_type == b'.': # Verifica si la entrada es un archivo.
                # Extrae el nombre, tamaño y clúster inicial del archivo.
                name, size, start_cluster = (
                    entry[1:16].decode('ascii').rstrip(),
                    struct.unpack('<I', entry[16:20])[0],
                    struct.unpack('<I', entry[20:24])[0]
                )
                if name.strip() == filename.strip(): # Comprueba si el nombre coincide con el archivo deseado.
                    img.seek(start_cluster * CLUSTER_SIZE) # Posiciona en el clúster inicial del archivo.
                    data = img.read(size) # Lee los datos del archivo.
                    with open(destination, 'wb') as dest_file: # Abre el archivo de destino en modo binario.
                        dest_file.write(data) # Escribe los datos en el archivo de destino.
                    return
    raise FileNotFoundError("Archivo no encontrado en FiUnamFS") # Error si no se encuentra el archivo.
```

La función `add_file(image_name, external_filename, internal_filename)` agrega un archivo desde el sistema local al FiUnamFS. Abre la imagen del sistema y el archivo local indicado por `external_filename`, para luego copiar y agregar el archivo dentro del sistema de archivos y posterior actualizar el directorio con el nombre `internal_filename`.

```

# Función para agregar un archivo desde el sistema local a FiUnamFS.
def add_file(image_path, source_path, dest_name):
    with open(image_path, 'r+b') as img: # Abre la imagen en modo lectura/escritura binaria.
        source_size = os.path.getsize(source_path) # Obtiene el tamaño del archivo origen.
        next_free_cluster = 5 # Asume que el siguiente clúster libre es el 5.
        free_entry_pos = None # Inicializa la posición de entrada libre como nula.

        img.seek(DIR_START) # Posiciona el cursor al inicio del directorio.
        for _ in range(DIR_SIZE // ENTRY_SIZE): # Itera sobre cada entrada del directorio.
            pos = img.tell() # Guarda la posición actual.
            entry = img.read(ENTRY_SIZE) # Lee una entrada completa.
            file_type = entry[0:1] # Tipo de archivo (byte 0).
            entry_cluster = struct.unpack('<I', entry[20:24])[0] # Extrae el clúster inicial.

            if file_type == b'#' and free_entry_pos is None:
                free_entry_pos = pos # Guarda la posición de la primera entrada libre.

            if entry_cluster >= next_free_cluster:
                next_free_cluster = entry_cluster + 1 # Calcula el siguiente clúster libre.

        if free_entry_pos is None:
            raise Exception("No hay espacio en el directorio") # Error si no hay entradas libres.

        with open(source_path, 'rb') as src_file: # Abre el archivo origen en modo binario.
            img.seek(next_free_cluster * CLUSTER_SIZE) # Posiciona en el clúster libre.
            img.write(src_file.read()) # Escribe el contenido del archivo en la imagen.

        img.seek(free_entry_pos) # Posiciona en la entrada libre encontrada.
        img.write(b'.' + dest_name.ljust(15).encode('ascii')) # Escribe el nombre del archivo en la entrada.
        img.write(struct.pack('<I', source_size)) # Guarda el tamaño del archivo.
        img.write(struct.pack('<I', next_free_cluster)) # Guarda el clúster inicial del archivo.

```

La función `delete_file(image_name, internal_filename)` elimina un archivo de FiUnamFS. Este busca dentro de FiUnamFS el archivo con el nombre dado por `internal_filename` en el directorio. Si el archivo es encontrado, en automático lo marca como eliminado y así libera el espacio que estaba ocupando dicho archivo.

```

# Función para eliminar un archivo en FiUnamFS.
def delete_file(image_path, filename):
    with open(image_path, 'r+b') as img: # Abre la imagen en modo lectura/escritura binaria.
        img.seek(DIR_START) # Posiciona el cursor al inicio del directorio.
        for _ in range(DIR_SIZE // ENTRY_SIZE): # Itera sobre cada entrada del directorio.
            pos = img.tell() # Guarda la posición actual.
            entry = img.read(ENTRY_SIZE) # Lee una entrada completa.
            name = entry[1:16].decode('ascii').rstrip() # Extrae y decodifica el nombre del archivo.
            if name.strip() == filename.strip(): # Comprueba si el nombre coincide con el archivo a eliminar.
                img.seek(pos) # Posiciona en la entrada del archivo.
                img.write(b'#' + b' ' * 15) # Marca la entrada como eliminada.
                print("Archivo eliminado") # Informa que el archivo fue eliminado.
                return
        raise FileNotFoundError("Archivo no encontrado en FiUnamFS") # Error si no se encuentra el archivo.

```

La función `defragment_fs(image_name)` va a reorganizar los archivos dentro de FIUnamFS para eliminar la fragmentación. Esto con el propósito de reorganizar el espacio disponible dentro del directorio y eliminando los huecos vacíos entre los archivos, para posterior escribir los datos ya organizados y actualizar el directorio.

```

# Función para desfragmentar el sistema de archivos.
def defragment_fs(image_path):
    entries = [] # Lista para almacenar entradas válidas.
    with open(image_path, 'r+b') as img: # Abre la imagen en modo lectura/escritura binaria.
        img.seek(DIR_START) # Posiciona el cursor al inicio del directorio.
        for _ in range(DIR_SIZE // ENTRY_SIZE): # Itera sobre cada entrada del directorio.
            entry = img.read(ENTRY_SIZE) # Lee una entrada completa.
            if entry[0:1] != b'#' and entry[1:16].strip(b' '):
                entries.append(entry) # Almacena entradas válidas en la lista.

    entries.sort(key=lambda e: struct.unpack('<I', e[20:24])[0]) # Ordena las entradas por clúster inicial.
    current_cluster = 5 # Define el clúster inicial para desfragmentar.
    for entry in entries:
        name, size, start_cluster = (
            entry[1:16].decode('ascii').rstrip(),
            struct.unpack('<I', entry[16:20])[0],
            struct.unpack('<I', entry[20:24])[0]
        )

        img.seek(start_cluster * CLUSTER_SIZE) # Posiciona en el clúster inicial.
        data = img.read(size) # Lee los datos del archivo.
        img.seek(current_cluster * CLUSTER_SIZE) # Posiciona en el nuevo clúster.
        img.write(data) # Escribe los datos en el clúster actual.

        new_entry = (
            entry[0:20] + struct.pack('<I', current_cluster) + entry[24:]
        )
        img.seek(DIR_START + ENTRY_SIZE * entries.index(entry))
        img.write(new_entry)
        current_cluster += (size + CLUSTER_SIZE - 1) // CLUSTER_SIZE # Avanza al siguiente clúster disponible.

    print("Desfragmentación completada.") # Mensaje de finalización.

```

Finalmente, la función `main_menu()` va a tener una interacción con le usuario, mostrando la interfaz con la que interactuara el usuario. Cuando se ejecute el programa imprimirá el menú con las opciones disponibles. El usuario selecciona una opción, y la función correspondiente es llamada para realizar la operación solicitada.

```

# Menú principal para interacción con el usuario.
def main_menu():
    # Llamada a la función datos del autor
    presentar_datos()
    while True:
        print("\nMenú Principal - Sistema de Archivos FiUnamFS")
        print("1. Leer el superbloque")
        print("2. Listar directorio")
        print("3. Copiar archivo de FiUnamFS a sistema")
        print("4. Copiar archivo de sistema a FiUnamFS")
        print("5. Eliminar archivo de FiUnamFS")
        print("6. Salir")
        choice = input("Selecciona una opción: ") # Solicita la opción del usuario.

        if choice == '1':
            read_superblock(image_path)
        elif choice == '2':
            list_directory(image_path)
        elif choice == '3':
            file_name = input("Nombre del archivo en FiUnamFS: ")
            dest_path = input("Ruta de destino en el sistema (dejar en blanco para carpeta actual): ")
            if not dest_path:
                dest_path = os.path.join(os.getcwd(), file_name)
            try:
                extract_file(image_path, file_name, dest_path)
                print("Archivo copiado con éxito a", dest_path)
            except Exception as e:
                print("Error al copiar el archivo:", e)
        elif choice == '4':
            src_path = input("Ruta del archivo en el sistema: ")
            dest_name = input("Nombre del archivo en FiUnamFS: ")
            add_file(image_path, src_path, dest_name)
        elif choice == '5':
            file_name = input("Nombre del archivo a eliminar de FiUnamFS: ")
            delete_file(image_path, file_name)
        elif choice == '6':
            print("Saliendo del programa.")
            break
        else:
            print("Opción no válida. Por favor, intenta de nuevo.")

image_path = "fiunamfs.img" # Ruta de la imagen de sistema de archivos.
main_menu() # Llama al menú principal para iniciar el programa.

```

Instrucciones para el uso del programa:

- Instalar cualquier versión de python 3.x
- Descargar el código proporcionado en este caso "Aquino_GabrielaP2.py"
- Copiar o abrir el código desde un IDE, puede ser Visual Studio Code.
- Una vez abierto o copiado el código, abre la terminal y ejecuta el siguiente comando si estas en Ubuntu 'python3 Aquino_GabrielaP2.py' o 'python Aquino_GabrielaP2.py' si estas en Windows.
- Una vez ejecutado te saldrá un menú como este:


```
Nombre: Gabriela Aquino Lozada
Facultad: Facultad de Ingeniería
Carrera: Ingeniería en Computación

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción:
```

- Selecciona la opción que deseas ejecutar del programa, ingresando el numero de la tarea a realizar.

Ejecución del programa:

Cuando el código se ejecute muestra la siguiente interfaz:

 C:\Users\gabir\AppData\Local\Programs\Python\Launcher\py.exe

```
Nombre: Gabriela Aquino Lozada
Facultad: Facultad de Ingeniería
Carrera: Ingeniería en Computación

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción: _
```

- Cuando se elige la opción 1:

```

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción: 1
Sistema de archivos: FiUnamFS, Versión: 25-1
Superbloque válido

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción:

```

El programa muestra que la validación del archivo leído es correcta, ya que se verificó que el nombre y la versión del sistema fueran las correctas.

- **Cuando se escoge la opción 2:**

```

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción: 2
Archivo: README.org
Archivo: logo.png
Archivo: mensaje.jpg

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción:

```

El programa muestra los archivos que se encuentran en FiUnamFs, al terminar de ejecutar esa tarea nos vuelve a imprimir el menú principal.

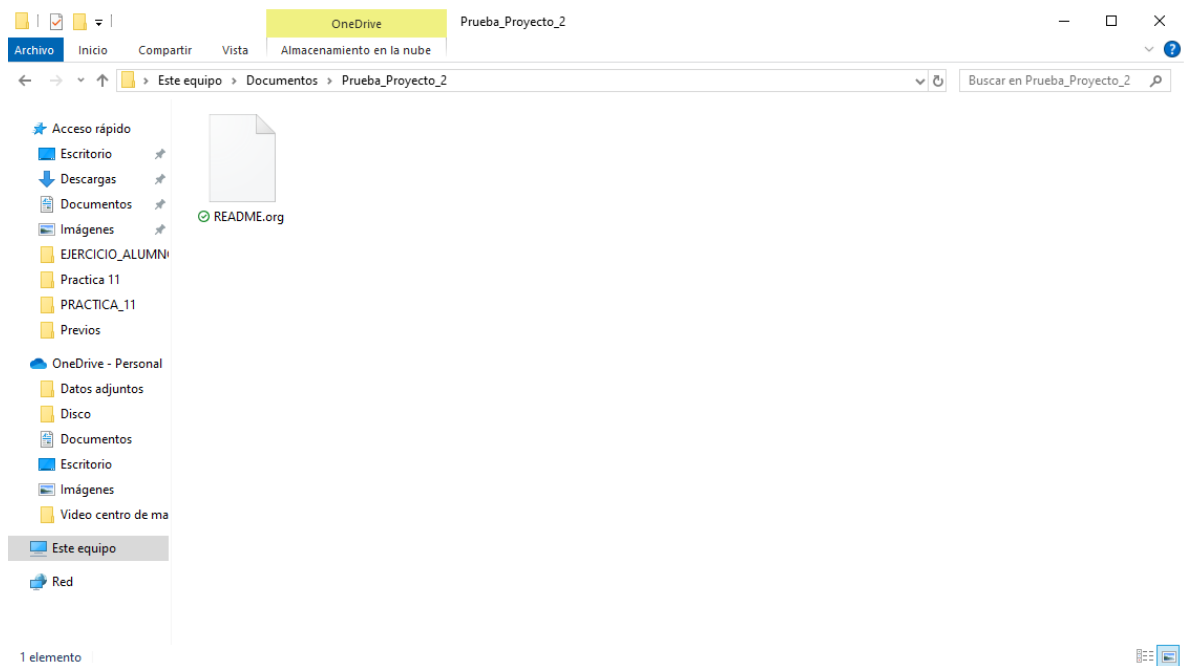
- **Cuando se escoge la opción 3:**

```

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción: 3
Nombre del archivo en FiUnamFS: README.org
Ruta de destino en el sistema (dejar en blanco para carpeta actual): C:\Users\gabir\OneDrive\Documentos\Prueba_Proyecto_2\README.org
Archivo copiado con éxito a C:\Users\gabir\OneDrive\Documentos\Prueba_Proyecto_2\README.org

Menú Principal - Sistema de Archivos FiUnamFS
1. Leer el superbloque
2. Listar directorio
3. Copiar archivo de FiUnamFS a sistema
4. Copiar archivo de sistema a FiUnamFS
5. Eliminar archivo de FiUnamFS
6. Salir
Selecciona una opción:

```



El programa te pide el nombre del archivo y la ruta donde quieres guardar el archivo y procede a almacenarlo en el equipo: En la ruta de la computadora muestra el archivo seleccionado en este caso README.org