Facultad de Ingeniería-UNAM

Nombre de la materia
Sistemas Operativos

Semestre: 2023-2

Prof: ING. Gunnar Eyal Wolf Iszaevich

Título del Trabajo.
Micro sistema de archivos

Número del trabajo.
Proyecto 3

Nombre del estudiante.

Miranda Barajas Victor
Grupo
06

Fecha de entrega.
25/Noviembre/ 2023

Breve resumen:

El programa FiUnamFS, fue desarrollado en Python, está diseñado para interactuar con un sistema de archivos personalizad. Ofrece funcionalidades como listar archivos en FiUnamFS, copiar archivos entre el sistema de archivos y el sistema local, y eliminar archivos de FiUnamFS. También incluye una función para desfragmentar el sistema de archivos, mejorando su eficiencia. Este software utiliza módulos estándar de Python, como os y struct, garantizando una amplia compatibilidad. Su propósito principal es proporcionar una herramienta de manejo sencillo y efectivo para operaciones básicas de archivos en FiUnamFS.

Características del Programa:

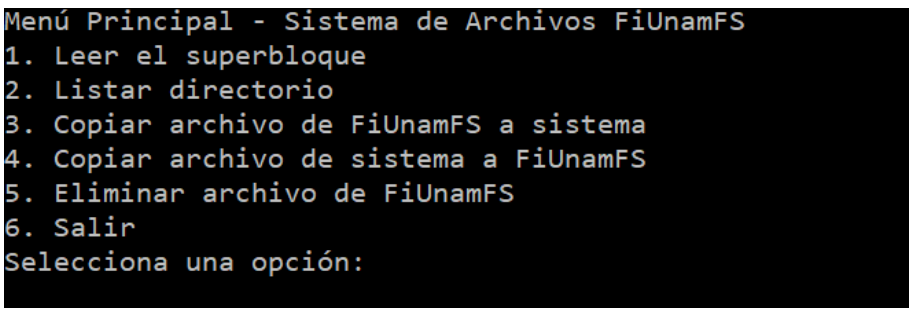
- Lenguaje de programación: Python, versión: 3.10.9
- Módulos Requeridos:

  os: Utilizado para interactuar con el sistema operativo.

  struct: Utilizado para trabajar con datos binarios.
- 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

Instrucciones de uso:

- Instalar python 3.x
- Descargar el código proporcionado en este caso “MirandaVictorP3.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 MirandaVictorP3.py’ o ‘python MirandaVictor.py ‘ si estas en Windows.
- Una vez ejecutado te saldrá un menú como este:



- Selecciona la opción que quieras ejecutar del programa de acuerdo con el número asignado a su izquierda.

Código en python comentado:

```
# Miranda Barajas Victor
import os
import struct


TAMANO_CLUSTER = 1024
TAMANO_ENTRADA = 64
DIRECTORIO_INICIO = TAMANO_CLUSTER  # Cluster 1
DIRECTORIO_TAMANO = 4 * TAMANO_CLUSTER  # 4 clusters para el directorio
MAXIMO_CLUSTERS = 1440 // 4  # Asumiendo que el tamaño total es 1440KB

# Función para leer y validar el superbloque del sistema de archivos.
def leer_superbloque(fiunamfs_img):
    with open(fiunamfs_img, 'rb') as f:
        f.seek(0)
        nombre_fs = f.read(8).decode('ascii').strip()
        f.seek(10)
        version = f.read(5).decode('ascii').rstrip('\x00').strip()
        print(f"Nombre FS leído: {nombre_fs}, Versión leída: {version}")
        #print(f"Hexadecimal: {nombre_fs.encode('ascii').hex()},
{version.encode('ascii').hex()}")
        if nombre_fs != "FiUnamFS" or version.strip() != "24.1":
            print("Valores no coinciden, se esperaba FiUnamFS y  24.1")
            #raise ValueError("No es un sistema FiUnamFS válido o versión no soportada.")
        else:
            print("Superbloque válido")
# Función para listar los archivos en el directorio de FiUnamFS.
def listar_directorio(fiunamfs_img):
    with open(fiunamfs_img, 'rb') as f:
        f.seek(DIRECTORIO_INICIO)
         # Iterar sobre cada entrada del directorio.
        for _ in range(DIRECTORIO_TAMANO // TAMANO_ENTRADA):
            entrada = f.read(TAMANO_ENTRADA)
            nombre = entrada[1:16].decode('ascii').rstrip()
             # Imprimir nombres de archivos válidos.
            if nombre != '-' * 15:
                print(f"Archivo: {nombre}")
# Función para copiar un archivo de FiUnamFS al sistema local.
def copiar_a_sistema(fiunamfs_img, nombre_archivo, destino):
    with open(fiunamfs_img, 'rb') as f:
        f.seek(DIRECTORIO_INICIO)
          # Buscar el archivo en el directorio.
        for _ in range(DIRECTORIO_TAMANO // TAMANO_ENTRADA):
            entrada = f.read(TAMANO_ENTRADA)
            tipo_archivo = entrada[0:1]
            if tipo_archivo == b'-':
                nombre, tam, cluster_ini = (
                    entrada[1:16].decode('ascii').rstrip(),
                    struct.unpack('<I', entrada[16:20])[0],
                    struct.unpack('<I', entrada[20:24])[0]
                )
                print(f"Nombre encontrado: {nombre}, Tamaño: {tam}, Cluster inicial:
{cluster_ini}")  # Para depuración
                if nombre.rstrip('\x00').strip() == nombre_archivo.rstrip('\x00').strip():
```

```
                    f.seek(cluster_ini * TAMANO_CLUSTER)
                    datos = f.read(tam)
                    with open(destino, 'wb') as archivo_destino:
                        archivo_destino.write(datos)
                    return
    raise FileNotFoundError("Archivo no encontrado en FiUnamFS")
# Función para copiar un archivo del sistema local a FiUnamFS
def copiar_a_fiunamfs(fiunamfs_img, archivo_origen, nombre_destino):
    with open(fiunamfs_img, 'r+b') as f:
        tam_origen = os.path.getsize(archivo_origen)
        cluster_libre = 5
        posicion_entrada_libre = None
         # Buscar una entrada vacía y un cluster libre.
        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:  # Verifica si la
entrada está vacía
                posicion_entrada_libre = posicion_actual
                print(f"Encontrada entrada libre en posición {posicion_entrada_libre}")  # Para
depuración

            if cluster_ini >= cluster_libre:
                cluster_libre = cluster_ini + 1
                print(f"Nuevo cluster libre: {cluster_libre}")  # Para depuración

        if posicion_entrada_libre is None:
            raise Exception("No hay espacio en el directorio")
        else:
            print(f"Espacio libre en la entrada: {posicion_entrada_libre}, Cluster libre para el
archivo: {cluster_libre}")  # Para depuración

        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))

#Función para eliminar archivos de FiUnamFS
def eliminar_archivo(fiunamfs_img, nombre_archivo):
    with open(fiunamfs_img, 'r+b') as f:
        f.seek(DIRECTORIO_INICIO)
        for _ in range(DIRECTORIO_TAMANO // TAMANO_ENTRADA):
            posicion = f.tell()
            entrada = f.read(TAMANO_ENTRADA)
            nombre = entrada[1:16].decode('ascii').rstrip()
            if nombre.rstrip('\x00').strip() == nombre_archivo.rstrip('\x00').strip():
                f.seek(posicion)
```

```
                f.write(b'/' + b' ' * 15)
                print("Archivo eliminado")
                return
    raise FileNotFoundError("Archivo no encontrado en FiUnamFS")
#Función para la desfragmentación de FiUnamFS
def desfragmentar(fiunamfs_img):

    # Leer todo el directorio
    entradas = []
    with open(fiunamfs_img, 'r+b') as f:
        f.seek(DIRECTORIO_INICIO)
        for _ in range(DIRECTORIO_TAMANO // TAMANO_ENTRADA):
            entrada = f.read(TAMANO_ENTRADA)
            if entrada[0:1] != b'/' and entrada[1:16].strip(b' ') != b'':
                entradas.append(entrada)

    # Ordenar las entradas por cluster inicial
    entradas.sort(key=lambda e: struct.unpack('<I', e[20:24])[0])

    # Reescribir los archivos de forma contigua
    cluster_actual = 5  # Comenzar después del directorio
    for entrada in entradas:
        nombre, tam, cluster_ini = (
            entrada[1:16].decode('ascii').rstrip(),
            struct.unpack('<I', entrada[16:20])[0],
            struct.unpack('<I', entrada[20:24])[0]
        )

        # Leer datos del archivo original
        f.seek(cluster_ini * TAMANO_CLUSTER)
        datos = f.read(tam)

        # Escribir datos en la nueva ubicación
        f.seek(cluster_actual * TAMANO_CLUSTER)
        f.write(datos)

        # Actualizar entrada del directorio
        nueva_entrada = (
            entrada[0:20] + struct.pack('<I', cluster_actual) + entrada[24:]
        )
        f.seek(DIRECTORIO_INICIO + TAMANO_ENTRADA * entradas.index(entrada))
        f.write(nueva_entrada)

        # Actualizar el siguiente cluster libre
        cluster_actual += (tam + TAMANO_CLUSTER - 1) // TAMANO_CLUSTER  # Redondear hacia arriba

    print("Desfragmentación completada.")
def menu_principal():
    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")
        opcion = input("Selecciona una opción: ")

        if opcion == '1':
            leer_superbloque(fiunamfs_img)
        elif opcion == '2':
            listar_directorio(fiunamfs_img)

        elif opcion == '3':
            nombre_archivo = input("Ingresa el nombre del archivo en FiUnamFS: ")
            destino = input("Ingresa la ruta de destino en el sistema (deja en blanco para la
carpeta actual): ")
            if not destino:
                destino = os.path.join(os.getcwd(), nombre_archivo)
            try:
                copiar_a_sistema(fiunamfs_img, nombre_archivo, destino)
                print("Archivo copiado con éxito a", destino)
            except Exception as e:
                print("Error al copiar el archivo:", e)

        elif opcion == '4':
            origen = input("Ingresa la ruta del archivo en el sistema: ")
            nombre_destino = input("Ingresa el nombre del archivo en FiUnamFS: ")
            copiar_a_fiunamfs(fiunamfs_img, origen, nombre_destino)
        elif opcion == '5':
            nombre_archivo = input("Ingresa el nombre del archivo a eliminar de FiUnamFS: ")
            eliminar_archivo(fiunamfs_img, nombre_archivo)
        elif opcion == '6':
            print("Saliendo del programa.")
            break
        else:
            print("Opción no válida. Por favor, intenta de nuevo.")

# Configuración inicial
fiunamfs_img = "fiunamfs.img"

# Ejecutar el menú principal
menu_principal()
```

Conclusión:

El proyecto de “Micro sistema de archivos” representa un ejemplo sobresaliente de cómo se pueden aplicar los principios de la informática y la programación para crear soluciones prácticas y funcionales. A través de este proyecto, hemos explorado aspectos fundamentales del manejo de sistemas de archivos, incluyendo la lectura y escritura de datos en un formato definido, la manipulación de archivos y la desfragmentación.