UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

SISTEMAS OPERATIVOS

Grupo 6

PROYECTO NO. 4

(Micro) Sistema de Archivos

ALUMNOS

Abad Vásquez, Aldo

Rosales López, Luis André

PROFESOR

Ing. Gunnar Eyal Wolf Iszaevich

Ciudad Universitaria, Cd. Mx., 19 de mayo de 2022

Proyecto 3: Asignación de Memoria a un Sistema Real

Introducción

De los papeles que cumple el sistema operativo, probablemente el que más consciente tengan a sus usuarios es el de la gestión del espacio de almacenamiento, esto es, la organización de la información en un sistema de archivos.

La información cruda tiene que pasar una serie de transformaciones, yendo de niveles superiores a los más bajos. Un programa estructura sus datos en archivos, siguiendo el formato que resulte más pertinente al tipo de información a representar y un conjunto de archivos hoy en día es típicamente representado en una estructura de directorios.

Sin embargo, los archivos son una estructura meramente lógica, por lo que deben ser convertidos para ser representados en un dispositivo de bloques como los diversos tipos de unidades como discos.

Planteamiento del proyecto

Desarrollar es un programa que pueda obtener, crear y modificar información en el microsistema de archivos de la Facultad de Ingeniería, ***FiUnamFS.***

Siguiendo la especificación que aparece en la siguiente sección, tienen que desarrollar un programa que pueda:

1. Listar los contenidos del directorio
2. Copiar uno de los archivos de dentro del FiUnamFS hacia tu sistema
3. Copiar un archivo de tu computadora hacia tu FiUnamFS
4. Eliminar un archivo del FiUnamFS
5. Dado que este sistema de archivos es dado a la fragmentación externa es necesario también un programa que desfragmente al FiUnamFS.

Especificación de FiUnamFS

- El sistema de archivos cabe en un *diskette* tradicional. Claro, no espero que tengan acceso al hardware, por lo que lo simularemos representándolo en un archivo de longitud fija, de 1440 Kilobytes.
- Por simplicidad, si bien en un sistema de archivos real representaríamos las magnitudes en su representación binaria, vamos a hacerlo con cadenas (esto es, en vez de representar al número 1354 como los caracteres ASCII 0, 0, 5, 74 (porque 5 * 256 = 1280, 1280 + 74 = 1354), represéntala como la cadena 1354. Si el campo en que vas a ubicar este valor mide más del espacio necesario, agrega ceros a la izquierda.
- La superficie del disco se divide en sectores de 256 bytes. Cada *cluster* mide cuatro sectores.
- El pseudo dispositivo no maneja *tabla de particiones*, hospeda directamente un volumen en la totalidad de su espacio.
- *FiUnamFS* maneja únicamente un directorio plano, no se consideran subdirectorios.
- El primer *cluster* (#0) del pseudo dispositivo es el super bloque. Este contiene información en los siguientes bytes:

  0–8 Para identificación, el nombre del sistema de archivos. ¡Debes validar nunca modificar un sistema de archivos que no sea el correcto! Debe ser la cadena FiUnamFS.
  10–13 Versión de la implementación. Estamos implementando la versión 1.1
  20–35 Etiqueta del volumen
  40–45 Tamaño del cluster en bytes
  47–49 Número de clusters que mide el directorio
  52–60 Número de clusters que mide la unidad completa

- El resto del super bloque puede quedar vacío (o puedes sobrecargarlo con otros datos)
- El sistema de archivos es de asignación contigua. Toda la información de los archivos está en el directorio.
- El directorio está ubicado en los *clusters* 1 a 4. Cada entrada del directorio mide 64 bytes, consistentes en:

0–15 Nombre del archivo

16–24 Tamaño del archivo, en bytes

25–30 *Cluster* inicial

31–45 Hora y fecha de creación del archivo, especificando AAAAMMDDHHMMSS (p.ej. '20220508182600' para 2022-05-08 18:26:00)

46–60 Hora y fecha de última modificación del archivo, especificando AAAAMMDDHHMMSS (p.ej. '20220509182600')

61–64 Espacio no utilizado (¿reservado para expansión futura?)

- Las entradas no utilizadas del directorio se identifican porque en el campo de nombre llevan la cadena .....
- Los nombres de archivos pueden componerse de cualquier caracter dentro del subconjunto ASCII de 7 bits (no acentuados, no Unicode, sólo el viejo y aburrido US-ASCII)
- Es un sistema de archivos plano — No maneja subdirectorios.
- Después del directorio, todo el espacio restante es espacio de datos.

Implementación

Descripción del Entorno de Desarrollo

Lenguaje y versión utilizados. El lenguaje empleado para la implementación es Python 3 (3.7.3). Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en una sintaxis que favorezca un código legible. Se trata de un lenguaje de programación multiparadigma, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y es multiplataforma.

Bibliotecas utilizadas

- **Lib/os.py** : este módulo provee una manera versátil de usar funcionalidades dependientes del sistema operativo. Si quieres leer o escribir un archivo usa open(), si quieres manipular rutas, usa el módulo os.path, y si quieres leer todas las líneas en todos

los archivos en la línea de comandos usa el módulo fileinput. Para crear archivos temporales y directorios usa el módulo tempfile, y para el manejo de alto nivel de archivos y directorios puedes usar el módulo shutil.

- **sys** : este módulo provee acceso a algunas variables usadas o mantenidas por el intérprete y a funciones que interactúan fuertemente con el intérprete. Siempre está disponible..
- **Lib/struct.py** : este módulo realiza conversiones entre valores de Python y estructuras C representadas como objetos de bytes de Python. Esto se puede usar para manejar datos binarios almacenados en archivos o de conexiones de red, entre otras fuentes. Utiliza cadenas de formato como descripciones compactas del diseño de las estructuras C y la conversión prevista a/desde valores de Python.
- **Lib/getpass.py** : de este modulo se toma la siguiente función.
  - **getpass.getuser()** : devuelve el «nombre de inicio de sesión» del usuario.

Sistema operativo / distribución de desarrollo y/o pruebas. El desarrollo se realizó y probó en en Windows 11 mediante el *IDLE* de *Python*. Es importante que el usuario que desee utilizar el programa se asegure de tener instalado el intérprete de Python (Python3) así como las bibliotecas mencionadas anteriormente. Esto se puede hacer directamente, en Linux, con el comando:

```
$ pip install <biblioteca>
```

Después de esto es posible ejecutar el programa con el siguiente comando:

```
$ python3 fiunamfs.py
```

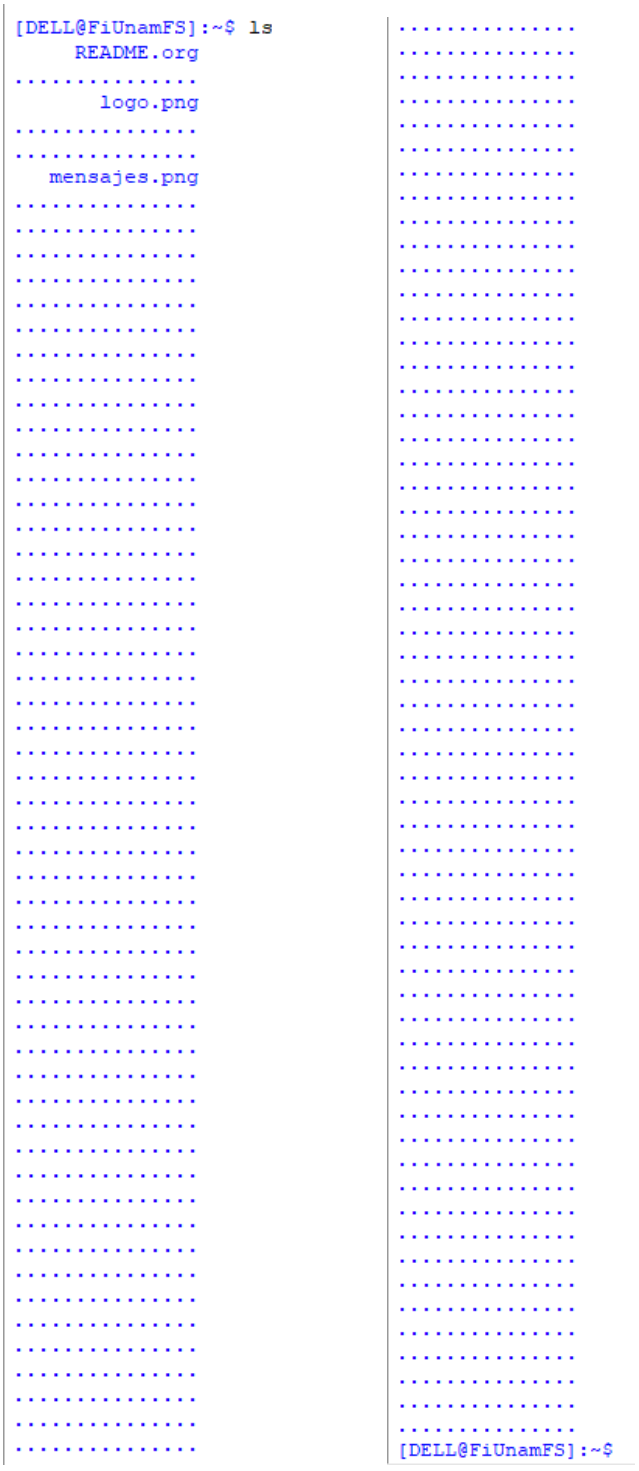
Adicionalmente, es importante que el archivo correspondiente al sistema de archivos se encuentre en la misma dirección que el archivo .py para su ejecución.

Evidencias de Ejecución

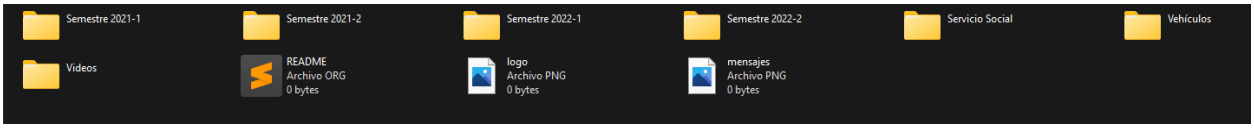
Una vez llegados a un punto donde el sistema pareciera medianamente funcional, se probo la ejecución del programa.

```
======== RESTART: D:\Semestre 2022-2\Sistemas Operativos\P4\fiunamfs.py ========
Use el comando 'help' para conocer el menú de comandos.

[DELL@FiUnamFS]:~$ help

 _ _ _ _ _ _ _ _ _ Comandos del Sistema _ _ _ _ _ _ _ _ _
| help                -> Desplegar menú de ayuda         |
| ls                  -> Listar contenido del directorio |
| cp «archivo» «ruta» -> Copiar archivo                  |
| rm «nombre_archivo» -> Eliminar archivo de FiUnamFS    |
| defrag              -> Desfragmentar FiUnamFS          |
| exit                -> Salir de FiUnamFS               |
|____________________________________________________|

[DELL@FiUnamFS]:~$
```

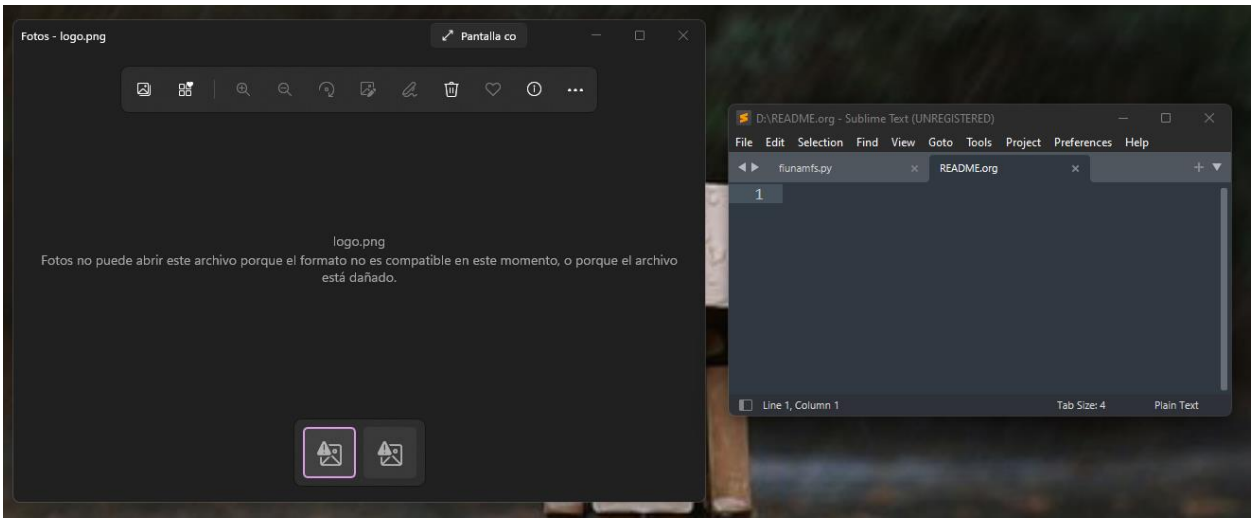
Primero con el comando *ls*, con lo cual se pudo observar que el sistema de archivos sólo contiene tres elementos.



Posteriormente se intentó copiar los archivos a nuestro sistema.

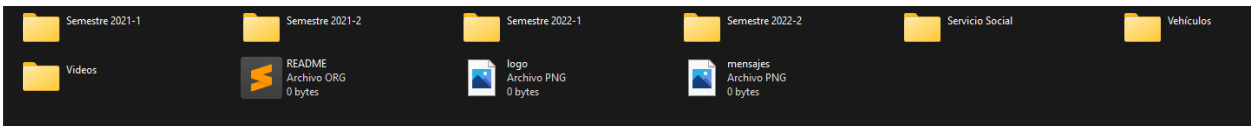


Sin embargo, al abrir los archivos ninguno contenía información.



Por ello, se intentó desfragmentar el sistema. Sin embargo, al volver a copiar los archivos, estos seguían vacíos.

```
[DELL@FiUnamFS]:~$ help

 _ _ _ _ _ _ _ _ _ Comandos del Sistema _ _ _ _ _ _ _ _ _

| help                -> Desplegar menú de ayuda          |

| ls                  -> Listar contenido del directorio  |

| cp «archivo» «ruta» -> Copiar archivo                   |

| rm «nombre_archivo» -> Eliminar archivo de FiUnamFS     |

| defrag              -> Desfragmentar FiUnamFS           |

| exit                -> Salir de FiUnamFS                |

|_____________________________________________________|

[DELL@FiUnamFS]:~$ cp README.org D:\
[DELL@FiUnamFS]:~$ cp logo.png D:\
[DELL@FiUnamFS]:~$ cp mensajes.png D:\
[DELL@FiUnamFS]:~$ defrag
[DELL@FiUnamFS]:~$ cp README.org D:\
[DELL@FiUnamFS]:~$ cp logo.png D:\
[DELL@FiUnamFS]:~$ cp mensajes.png D:\
[DELL@FiUnamFS]:~$
```



Llegados a este punto tenemos 3 hipótesis sobre lo ocurrido: la primera es que el sistema de archivos estuviera “corrupto” o de hecho los archivos estén vacíos, la segunda es que nuestro método para copiar de *FiUnamFS* a nuestro sistema esté mal implementada y la tercera es que nuestro método de desfragmentación sea incorrecto. Sobre la primera, podríamos comprobarla preguntándole al profesor. La segunda sería fácil de comprobar si tuviéramos implementado el método para copiar de nuestro sistema hacia *FiUnamFS*, sin embargo, ese no es el caso*.

*ACLARACIÓN IMPORTANTE: nuestra implementación sólo permite copiar archivos de *FiUnamFS* a nuestro sistema. Su gemela no fue implementada.

Finalmente, se probó exitosamente la eliminación de un archivo de *FiUnamFS*.

```
 _ _ _ _ _ _ _ _ _ Comandos del Sistema _ _ _ _ _ _ _ _ _
| help                -> Desplegar menú de ayuda         |
| ls                  -> Listar contenido del directorio |
| cp «archivo» «ruta» -> Copiar archivo                  |
| rm «nombre_archivo» -> Eliminar archivo de FiUnamFS    |
| defrag              -> Desfragmentar FiUnamFS          |
| exit                -> Salir de FiUnamFS               |
|____________________________________________________|

[DELL@FiUnamFS]:~$ rm logo.png
Archivo eliminado de FiUnamFS exitosamente.

[DELL@FiUnamFS]:~$ ls
     README.org
.................
.................
.................
   mensajes.png
.................
.................
.................
.................
```

Referencias

- Wolf, G., Ruiz, E., Bergero, F., & Meza, E. (2015). *Fundamentos de Sistemas Operativos*. Universidad Nacional Autónoma de México, Instituto de Investigaciones Económicas : Facultad de Ingeniería. http://sistop.org/pdf/sistemas_operativos.pdf