



**UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO  
FACULTAD DE INGENIERÍA**

**DEPARTAMENTO DE COMPUTACIÓN  
DIVISIÓN DE INGENIERÍA ELÉCTRICA  
SISTEMAS OPERATIVOS (2016)**



## **PROYECTO 3: MICRO-SISTEMA DE ARCHIVOS**

**ASIGNATURA: SISTEMAS OPERATIVOS (2016)**

**GRUPO: 4**

**PROFESOR: GUNNAR EYAL WOLF ISZAEVICH**

**ALUMNOS:**

**LÓPEZ SOTO MIGUEL ANGEL**

**DURÁN ROMERO JOSÉ ARTURO**

**23 / 04 / 2019**

## ➤ **Introducción:**

El proyecto presenta un programa que permite dado un archivo .img un sistema de archivos muy sencillo bajo las características que se solicitaron y que permite realizar las siguientes operaciones:

- Montaje
- Listado de archivos
- Copiar archivos al FS
- Copiar archivos hacia el FS
- Eliminar archivos
- Desfragmentar

## ➤ **Desarrollo**

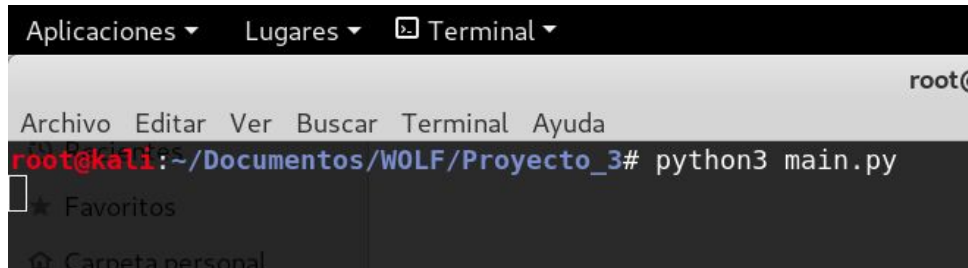
El programa está dividido en dos archivos: main.py contiene los métodos y procedimientos necesarios para implementar la interfaz gráfica. FS.py contiene la lógica correspondiente al sistema de archivos. A continuación se describe dicha lógica.

Se crearon dos clases una de tipo FileSystem que contiene parámetros de inicialización y los siguientes métodos:

- ❖ `_init_` : Inicialización
- ❖ `valida_FS`: Valida Si el nombre y la versión del sistema de archivos son correctos
- ❖ `getDataSB`: Obtiene los datos del primer clúster
- ❖ `getFilesInfo`: Obtiene los metadatos iniciando desde que termina el clúster 0 hasta el número de clusters del directorio
- ❖ `dataToFile`: Convierte los metadatos en un objeto tipo File para que la manipulación sea más sencilla. Contiene todos los atributos como nombre, número de bytes... etc
- ❖ `ls`: Lista los archivos
- ❖ `copytoUser`: Realiza la copia de un archivo a la computadora del usuario
- ❖ `getNamesFiles`: Obtiene los nombres de los archivos existentes
- ❖ `copyToFS`: Copia un archivo del usuario al FS
- ❖ `agregaMD`: Escribe los metadatos en el primer lugar que encuentra con la palabra "Aquí no hay nada"
- ❖ `dataToMD`: Cuando encuentra el espacio ahí guarda los datos
- ❖ `AvailableSpace`: Determina el espacio disponible en el FS
- ❖ `remove`: Función que elimina un archivo en el FS
- ❖ `desfragmentacion`: Para la desfragmentacion lo que se hace es acomodar todos los archivos existentes de manera contigua para así al escribir un archivo en el FS se escriba al final
- ❖ `getDataFromFileName`: Retorna los datos (bytes) de un archivo dado su nombre
- ❖ `setAtrb_MD`: Sobreescribe un atributo dado en los metadatos

### ➤ Ejecución:

La ejecución en consola es:



```
Aplicaciones ▾ Lugares ▾ Terminal ▾  
root@kali:~  
Archivo Editar Ver Buscar Terminal Ayuda  
root@kali:~/Documentos/WOLF/Proyecto_3# python3 main.py  
★ Favoritos  
📁 Carpeta personal
```

Y se muestra la ventana de control:



Como ventana emergente un aviso, pide que lo primero que se haga es montar el archivo .img

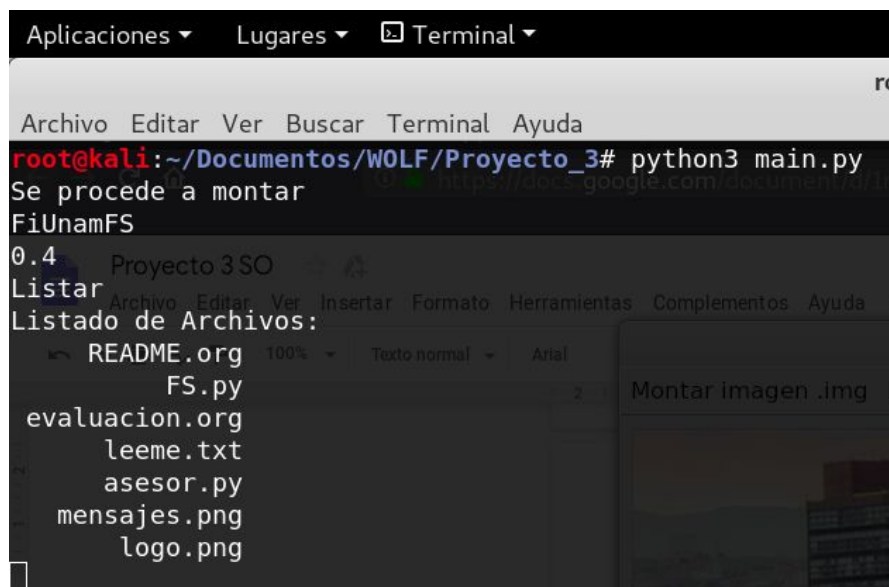
Se pulsa aceptar y podemos ver el menú en la parte superior. Damos click en montar imagen:



Se selecciona la imagen proporcionada en el proyecto:



En la consola podremos ver el resultado de cada acción, en este caso pulsamos en listar para ver el contenido de la imagen montada:



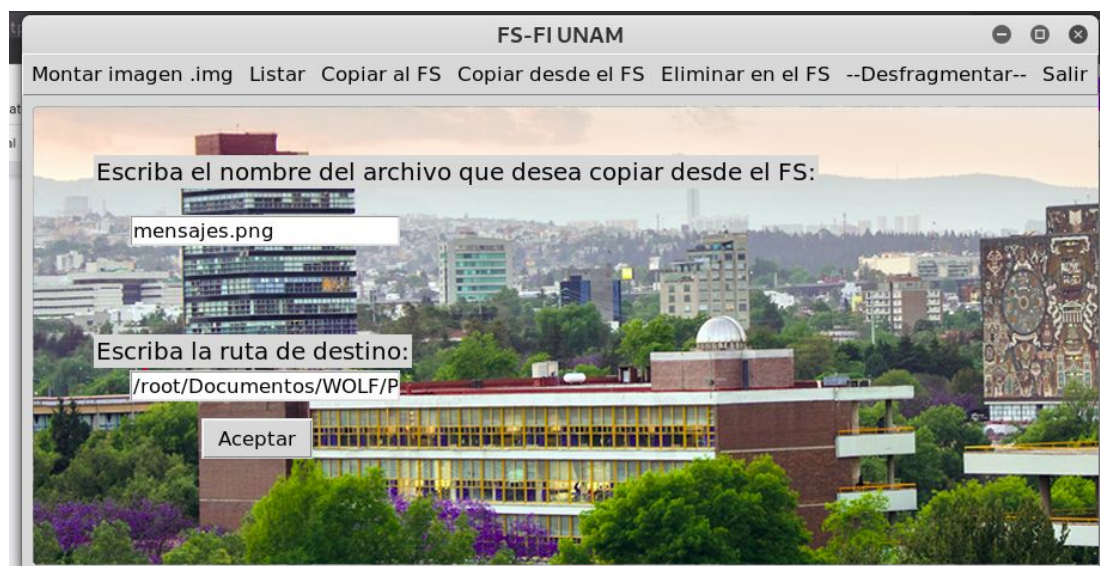
Ahora si deseamos copiar un archivo hacia el FS, damos click en la opción correspondiente en el menú y nuevamente una ventana emergente nos pedirá el archivo mismo que debemos seleccionar





En la consola podremos observar los resultados de cada operación

Para copiar desde el FS, tendremos la ventana siguiente:

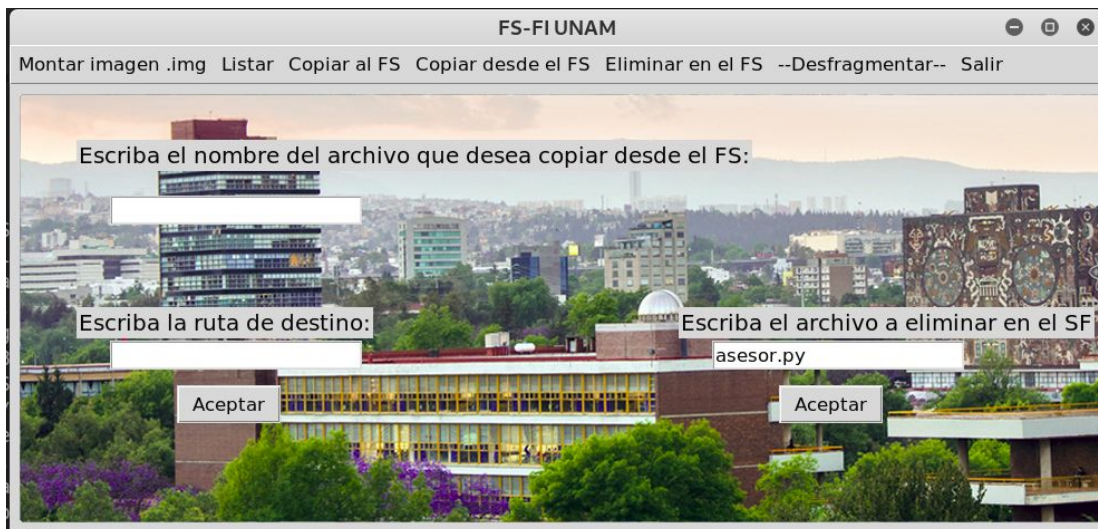


Debemos colocar en el cuadro el nombre del archivo que se copiará desde el FS. A continuación escribir la ruta completa de destino de dicho archivo, caso de algún error será mostrado en la consola.

En consola:

```
Se procede a copiar desde FS
Archivo a copiar: mensajes.png
Ruta: /root/Documentos/WOLF/Proyecto_3/Y/
Copiando ...
    mensajes.png
Listo!!!!
```

Para eliminar un archivo bastará con colocar el nombre del archivo:



En la consola podremos ver si hubo algún problema, para el caso no lo hubo:

```
Se procede a eliminar en el FS
Archivo a eliminar: asesor.py
{'README.org': 1024, 'FS.py': 1088, 'evaluacion.org': 1152, 'leeme.txt': 1216, 'asesor.py': 1280, 'mensajes.png': 1344, 'logo.png': 1408}
```

Al listar:

```
Listar
Listado de Archivos:
    README.org
    FS.py
    evaluacion.org
    leeme.txt
    mensajes.png
    logo.png
```

Observando la eliminación.

Si damos clic en desfragmentar, en consola se listan los archivos para verificar la integridad de los datos después de la desfragmentación.

```
Listado de Archivos:
  README.org
  FS.py
  evaluacion.org
  leeme.txt
  mensajes.png
  logo.png
Listado de Archivos:
  README.org
  FS.py
  evaluacion.org
  leeme.txt
  mensajes.png
  logo.png
```

Al dar click en salir el programa termina.

Contenido original del .img:



### ➤ **Condiciones de ejecución:**

El programa fue desarrollado utilizando el paradigma orientado a objetos la ejecución es mediante la ejecución de: **\$python3 main.py**

Versión y lenguaje de programación: Se desarrolló en y probó en **Python 3.7** **INDISPENSABLE USAR PYTHON 3.7.2** debido a que en Python 2.7 ocurren errores al montar el archivo .img

Para el desarrollo y pruebas

Existieron dos entornos:

1:

SO: Kali Linux

Programación: editor de texto Atom

Pruebas: ejecución por consola con Python 3.7

2:

SO: Windows 10

Programación y pruebas: IDE Pycharm community

Fin del documento.