



UNIVERSIDAD AUTONOMA DE
MÉXICO



FACULTAD DE INGENIERIA

SISTEMAS OPERATIVOS

**Proyecto 4. (Micro) Sistema de
Archivos**

Profesor: Gunnar Eyal Wolf Iszaevich

Integrantes:

Dávila Ortega Jesus Eduardo

Espinosa Cortez Giselle

Grupo: 06

Fecha de entrega: 19-05-2022

Contenido

INTRODUCCION	2
IMPLEMENTACIÓN	5
ESTRATEGIA.....	6
EXPLICACIÓN ACERCA DEL USO	6
Listar los contenidos del directorio	6
Copiar uno de los archivos de dentro del FiUnamFS hacia tu sistema	7
Copiar un archivo de tu computadora hacia tu FiUnamFS	9
Eliminar un archivo del FiUnamFS	10
Opción de desfragmentación	10
EJEMPLOS DE USO	12

INTRODUCCION

Para implementar nuestro sistema se utilizaron Seek(), Read() y Write(), durante el desarrollo de nuestra implementación dimos una nueva implementación de Read() llamada leer(), esta implementación nos permite establecer rangos de lectura así mismo establecemos checkpoints para poder regresar luego de haber leído las secciones que necesitamos.

En la implementación se utilizo la creación de una clase llamada sistema, la cual tiene como atributos las características de FiUnamFS, como la longitud de cluster, la cantidad de clusters que hay por directorio, la cantidad de clusters por unidad completa, así como su nombre, la versión de implementación y la etiqueta que posee el volumen.

class sistema:

```
#Atributos de nuestra clase sistema por defecto
file = open('fiunamfs.img',"br+")
nombre = ''
version = ''
etiquetaVol = ''
LongitudCluster = 0
cantidadClusterD = 0
cantidadClusterU = 0
lonitudEntradaDir = 64
```

Esta clase tiene métodos específicos de clase, a continuación, se hará mención de aquellos que se consideran de principal relevancia.

```

#Método encargado de añadir un archivo a FiUnamFs
def addArchivo(self):
    salida.delete('1.0',END)
    try:
        ruta = Ruta2.get()
    except ValueError:
        salida.insert(INSERT,'Valores Incorrectos')

    #Se busca que el archivo exista
    if os.path.isfile(ruta) == False:
        salida.insert(INSERT,"Error no existe el archivo origen: "+path)
        return
    #Obteniendo los datos del archivo y adecuandolos a nuestro formato
    tamaño = os.stat(ruta).st_size
    nombre = agregarEspacios(os.path.split(ruta)[-1])
    nombre = bytes(nombre,"utf-8")

    #Se busca observar si el nombre no excede nuestra especificación
    if nombre.__len__() > 15:
        salida.insert(INSERT,"Error nombre demasiado grande, solo se permiten 15 caracteres")
        return

    #Se procede a observar si no hay un archivo con el mismo nombre dentro de FiUnamFs
    if self.repetido(nombre) == True:
        salida.insert(INSERT,"Error ya existe un archivo con el mismo nombre")
        return

    #Obteniendo el tamaño del archivo y comprobando si cumple con nuestras reglas.
    size = ConversionBytes(tamaño)

```

El método addArchivo es un método de la clase sistema el cual es encargado de añadir archivos de nuestro sistema a FiUnamFS, para esto se realiza la comprobación de que el archivo cumpla como los requisitos, por ejemplo que su nombre no exceda los 15 caracteres y a su vez que se tenga suficiente espacio para poder almacenarlo, así mismo en este método también se auxilia de otros métodos para poder dar formato a todo lo que necesitamos para poder mover un archivo de nuestro sistema a FiUnamFS.

```

#Método que se encarga de copiar un archivo de FiUnamFs a nuestro equipo
def obtenerArchivo(self):
    salida.delete('1.0',END)
    try:
        nombre = Archivo1.get()
        ruta = Ruta1.get()
    except ValueError:
        salida.insert(INSERT,'Valores Incorrectos')
    try:
        ruta = Ruta1.get()
    except ValueError:
        salida.insert(INSERT,'Valores Incorrectos')
    #Se obtiene el nombre en un formato para facilitar la busqueda
    AuxNombre = bytes(agregarEspacios(nombre),"utf-8")
    tamaño = 0
    AlertaEncontrado = False

    #Se procede a buscar el archivo en caso de no encontrarlo se lanza el mensaje
    for i in range(self.LongitudCluster,(int(self.cantidadClusterD)+1)*self.LongitudCluster,self.lonitudEntradaDir):
        fila = self.leer(i, i + 15)
        if AuxNombre == fila:
            cluster = int(self.leer(i + 25,i + 30))
            tamaño = int(self.leer(i+16,i+24))
            AlertaEncontrado = True
            break

    if AlertaEncontrado == False:
        salida.insert(INSERT,"Error el archivo "+nombre+" no se encontro o no existe")
        return

```

El método obtenerArchivo es un método de la clase sistema como su nombre indica se encarga de obtener un archivo de FiUnamFS y moverlo a nuestro sistema especificando el lugar donde lo vamos

a almacenar, para estos se busca el nombre del archivo y una vez encontrado se lee su cluster y el tamaño, para así poder leer todo su contenido y almacenarlo, para posteriormente crear el archivo en nuestro sistema y escribir el contenido previamente almacenado.

```
#Método encargado de eliminar un archivo de FiUnamFs
def borrarArchivo(self):
    salida.delete('1.0',END)
    try:
        archivo = Archivo2.get()
    except ValueError:
        salida.insert(INSERT,'Valores Incorrectos')
    #Se empieza a buscar el archivo para poder eliminarlo
    nombre = agregarEspacios(archivo)
    for i in range(self.LongitudCluster,(int(self.cantidadClusterD)+1)*self.LongitudCluster,self.lonitudEntradaDir):
        auxiliar = self.leer(i, i + 15)
        if nombre == ConvertirAString(auxiliar):
            self.file.seek(i)
            self.file.write(voidRegister())
            salida.insert(INSERT,"El archivo "+archivo+" fue eliminado satisfactoriamente")
            return
    salida.insert(INSERT,"No se pudo eliminar el archivo "+archivo)
    return
```

El método borrarArchivo es un método de la clase sistema encargado de la eliminación de un archivo dentro de FiUnamFS, para esto se obtiene el nombre del archivo a eliminar y se le da formato agregándole espacios y así poder hacer un match con lo obtenido dentro del auxiliar y encontrar el archivo, en caso de encontrarse se rescribe en donde se encuentra con un registro vacío.

```
#Método encargado de desfragmentar FiUnamFs
def desfragmentacion(self):
    salida.delete('1.0',END)
    directorio = self.getDirectorio()

    #Obtenemos el directorio y ordenamos en base a su primer cluster
    for i in range(0,directorio.__len__()):
        for j in range(0,directorio.__len__()-i-1):
            auxiliar = directorio[j]
            directorio[j] = directorio[j + 1]
            directorio[j + 1] = auxiliar

    #Observamos si hay clusters vacios antes que ellos
    for i in range(0,directorio.__len__()):
        #Se ignoran los vacios
        if directorio[i][:15] == b'.....':
            continue

    #Obtenemos el estado de cada entrada del directorio
    vacios = self.clusterVacio()

    #Se empieza a buscar la entrada mas proxima para poder mover nuestros datos
    for j in range(int(directorio[i][25:30])-1,4,-1):
        if vacios[j] == False:
            ClusterInicial = j + 1
            tamaño = int(directorio[i][16:24])
            informacion = self.leer(int(directorio[i][25:30])*self.LongitudCluster,(int(directorio[i][25:30])*self.LongitudCluster)+tamaño)
            #Se busca y se escribe en donde se encontro un lugar disponible
            self.file.seek(ClusterInicial * self.LongitudCluster)
            self.file.write(informacion)
```

El método desfragmentación se encarga de desfragmentar FiUnamFS, lo que realiza es recorrer el siguiente archivo al registro vacío mas cercano del archivo anterior, para esto se obtiene el estado de todas las entradas de los directorios para saber en donde podemos escribir.

```

#Método para obtener todos los datos del directorio, incluyendo los espacios vacios
def getDirectorio(self):
    archivos = []
    for i in range(int(self.LongitudCluster),int(self.LongitudCluster)*(int(self.cantidadClusterD)+1),int(self.longitudEntradaDir)):
        archivos.append(self.leer(i,i+self.longitudEntradaDir))
    return archivos

#Método encargado de imprimir el contenido del directorio, ignorando las entradas no utilizadas del directorio.
def imprimirDirectorio(self):
    salida.delete('1.0',END)
    directorio = self.getDirectorio()
    auxiliar = ""
    directorioFinal = []
    for i in range(0,directorio.__len__()):
        auxiliar2 = ConvertirAString(directorio[i][:15])
        if "....." != auxiliar2:
            directorioFinal.append(auxiliar2)
    salida.insert(INSERT,"Los archivos dentro son: \n")
    for i in directorioFinal:
        salida.insert(INSERT,i+"\n")

```

Estos son dos métodos de la clase sistema ambos son utilizados para poder imprimir el contenido de los directorios, para esto se empieza a obtener todo el contenido de los directorios y posteriormente en imprimirDirectorio se ignoran aquellos que tengan por nombre la especificación de entrada sin utilizar del directorio.

```

def imprimirCaracteristicas(self):
    salida.delete('1.0',END)
    sistemaNombre = ConvertirAString(self.nombre)
    sistemaVersion = ConvertirAString(self.version)
    sistemaEtiqueta = ConvertirAString(self.etiquetaVol)
    longitudCluster = str(self.LongitudCluster)
    cantidadClusterDir= str(self.cantidadClusterD)
    cantidadClusterUnidad = str(self.cantidadClusterU)

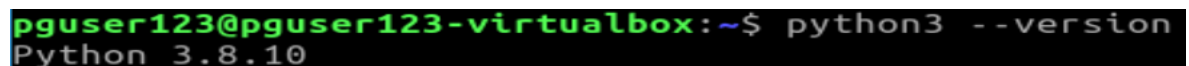
    salida.insert(INSERT,"Nombre del sistema de archivos: "+sistemaNombre+"\n")
    salida.insert(INSERT,"Version de implementacion: "+sistemaVersion+"\n")
    salida.insert(INSERT,"Etiqueta del volumen: "+sistemaEtiqueta+"\n")
    salida.insert(INSERT,"Longitud del cluster en bytes:"+longitudCluster+"\n")
    salida.insert(INSERT,"Numero de clusters que mide el directorio:"+cantidadClusterDir+"\n")
    salida.insert(INSERT,"Numero de clusters que mide la unidad completa:"+cantidadClusterUnidad+"\n")

```

El método imprimirCaracteristicas se encarga de obtener la cadena que se requiere para obtener los datos de FiUnamFS estas características son como su nombre, la versión de implementación, la etiqueta del volumen, la longitud del cluster, la cantidad de cluster que mide el directorio y la cantidad de clusters que mide la unidad completa. Si se requiere saber más de los métodos auxiliares que se utilizaron en la implementación se puede observar su función comentada en el código.

IMPLEMENTACIÓN

ENTORNO DE DESARROLLO



```

pguser123@pguser123-virtualbox:~$ python3 --version
Python 3.8.10

```

Python 3.8.10

Maquina Virtual de Ubuntu

Memory	2.9 GiB
Processor	Intel® Core™ i5-6440HQ CPU @ 2.60GHz
Graphics	llvmpipe (LLVM 12.0.0, 256 bits)
Disk Capacity	21.5 GB
OS Name	Ubuntu 20.04.4 LTS
OS Type	64-bit
GNOME Version	3.36.8
Windowing System	X11
Virtualization	Oracle

ESTRATEGIA

Uso de instrucciones Seek(), Read() y Write()

EXPLICACIÓN ACERCA DEL USO

Todas las funciones del sistema de archivos (Listas, borrar, exportar) son posibles tanto en WINDOWS como en LINUX. La desfragmentación solo se puede hacer en LINUX

Primero veremos que en la parte superior de la interfaz se encuentra un cuadro de texto que me pide insertar el dato necesario, esto es adicional para poder agregar el sistema de archivos que sea, pero en este caso da igual si ponemos el nombre del sistema de archivos FiUnamFS porque ya lo trae por defecto. Para que esta acción sea llevada a cabo exitosamente el sistema de archivos debe estar en la misma carpeta que el código.



Listar los contenidos del directorio

El botón listar archivos me permite listar los archivos dentro de FiUnamFs. Inicialmente no tenemos nada en el sistema de archivos:



Una vez que hemos agregado algo al sistema de archivos se ve de la siguiente manera:



Copiar uno de los archivos de dentro del FiUnamFS hacia tu sistema

Para poder realizar esta acción es necesario poner la dirección específica de donde el archivo se colocará y el nombre del archivo que queremos extraer del sistema de archivos en los cuadros de texto correspondientes, después de especificar lo mencionado anteriormente, dar clic en el siguiente botón:

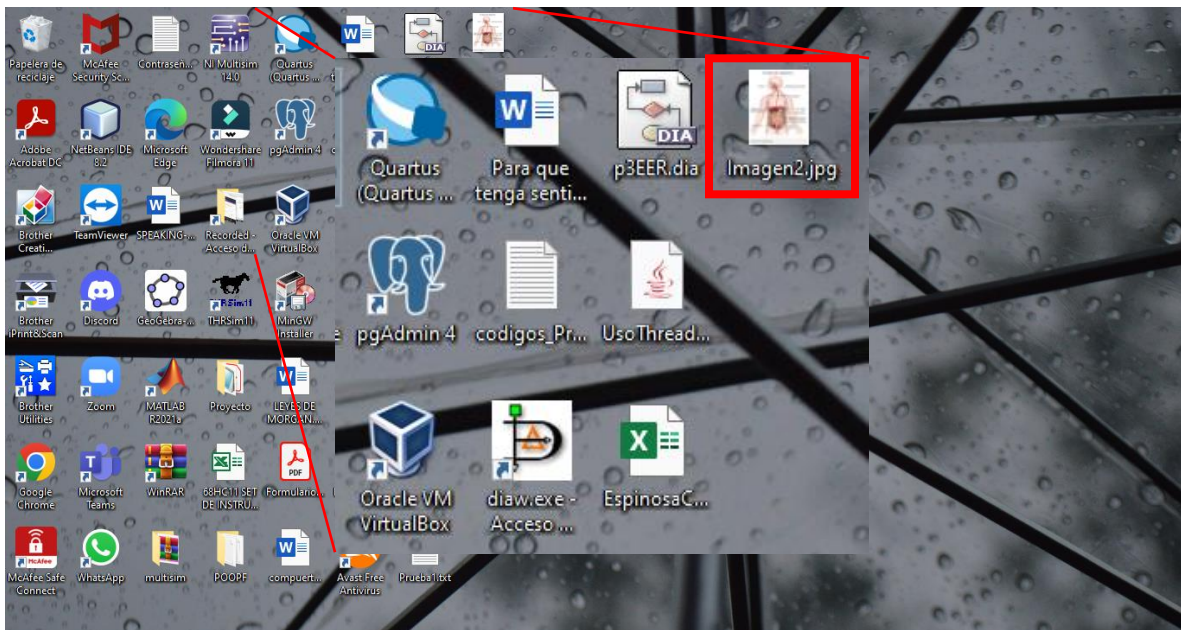
Copiar un archivo a tu sistema

Listar Archivos	
Copiar un archivo a tu sistema	
Inserte el nombre del archivo:	Imagen2.txt
Inserte la ruta donde se guardara el archivo:	C:\Users\DELL\Desktop
Copiar un archivo a FiUnamFs	
Inserte la ruta o el nombre del archivo:	
Desfragmentar	
Mapa de memoria disponible	
Borrar un archivo de FiUnamFs	
Inserte el nombre del archivo:	
Informacion de FiUnamFs	

El archivo Imagen2.jpg fue copiado exitosamente
en C:\Users\DELL\Desktop

Listar Archivos	
Copiar un archivo a tu sistema	
Inserte el nombre del archivo:	Imagen2.jpg
Inserte la ruta donde se guardara el archivo:	C:\Users\DELL\Desktop

Si revisamos nuestro escritorio el archivo fue copiado correctamente:



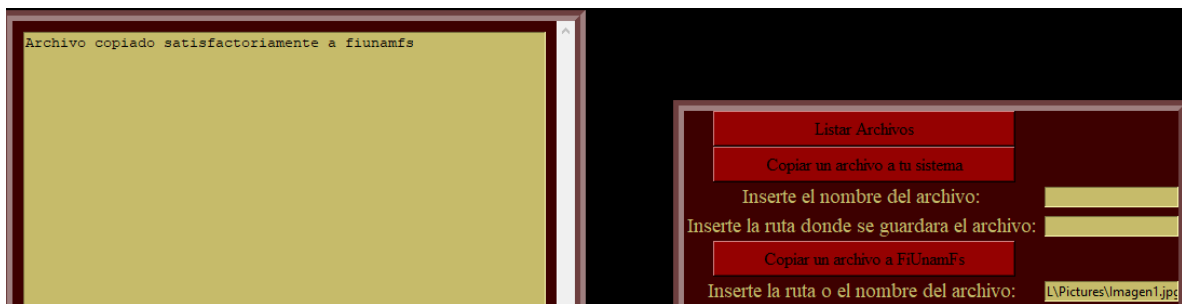
Copiar un archivo de tu computadora hacia tu FiUnamFS

Para insertar un documento es necesario tener la dirección exacta en la que se encuentra el archivo:

Colocar la ubicación en el cuadro de texto señalado y dar clic en el botón:



Recibiremos el siguiente mensaje si el archivo fue copiado correctamente:



Es importante mencionar que si el nombre es mayor a 15 caracteres no nos dejara insertar el archivo:



Eliminar un archivo del FiUnamFS

Inicialmente el listado de archivos se ve de la siguiente manera:

```
Los archivos dentro son:  
  Imagen1.jpg  
  Pruebal.txt  
CodigoP3DDM.txt  
  Imagen2.jpg  
  AparatoDig.jpg
```



Opción de desfragmentación

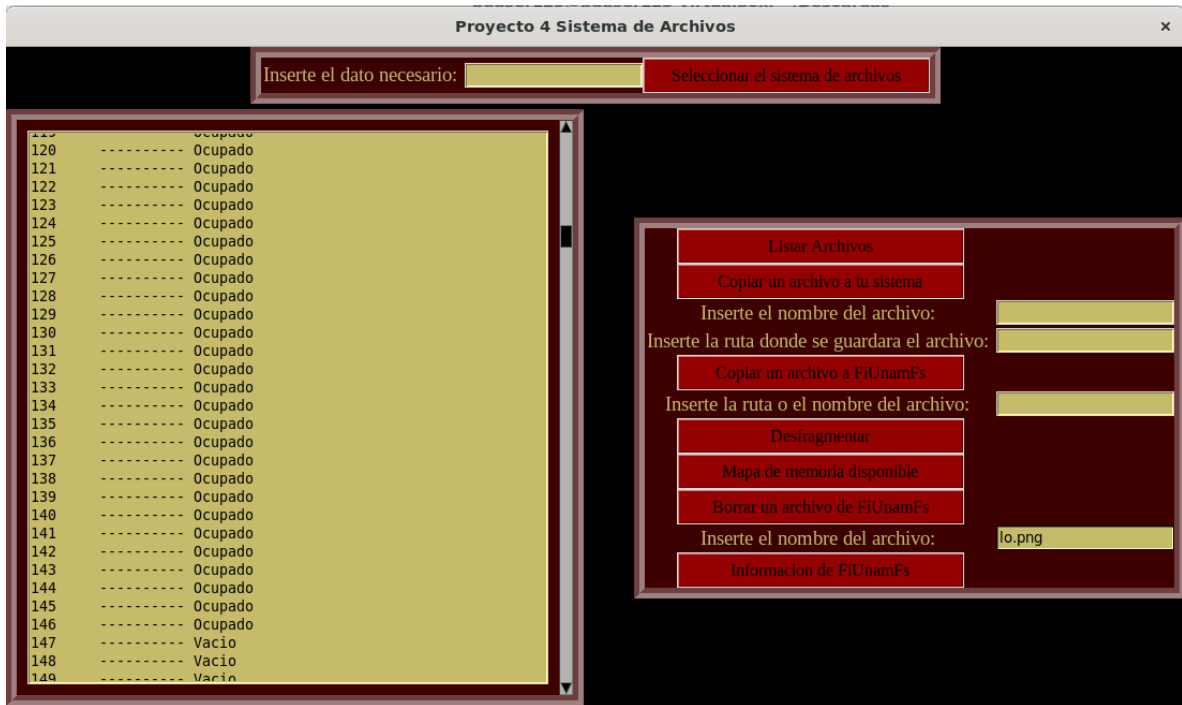
Al borrar algunos archivos hay que revisar el Portmap para ver si no hubo una fragmentación como vemos en el CLUSTER tenemos un ocupado y luego pasa a vacío.



Podríamos pensar que todo esta bien, pero si seguimos bajando vemos que hay se encuentra una combinación de ambos estados en diferentes intervalos por lo que debemos implementar el botón de desfragmentación para poder acomodar en el cluster de manera ordenada los espacios vacio y los ocupado.

Dando clic en el siguiente botón siguiente:





EJEMPLOS DE USO





