(Micro) sistema de archivos

REQUISITOS

Desarrollar un programa que pueda:

1. Listar los contenidos del directorio
2. Copiar uno de los archivos de dentro del FiUnamFS hacia nuestra computadora
3. Copiar un archivo de nuestra computadora hacia FiUnamFS
4. Eliminar un archivo de FiUnamFS
5. Generar un programa que desfragmente al FiUnamFS

LOGICA DE OPERACIÓN

Inicio del programa.

```
Imagen = open("fiunamfs.img", "r+b")
#Se hace el inicio del sistema leyendo los elementos almacenados en nuestro superbloque 0-57
Read = Imagen.read(57)
#Se reinicia el "apuntador" de nuestro disco
Imagen.seek(0)
#Se muestran los datos iniciales del sistema (Datos del superbloque)
print("          (Micro)Sistema de archivos:",Imagen.read(8).decode("latin_1"))
print("\nVersión: ", Imagen.read(10).decode("latin_1"), "\n")
print("Etiqueta volumen:", Imagen.read(20).decode("latin_1"), "\n")
#En el caso de los enteros al estar en little endian se obtienen los valores por medio del unpack
#regresando los valores en una tupla, por lo cual solo se almacena el elemento 0 de cada tupla
Size = unpack("i", Read[40:44])[0]
print("Tamaño cluster:", Size, "bytes\n")
ClusDirect = unpack("i", Read[45:49])[0]
print("Numero de clusters (directorio):", ClusDirect, "\n")
ClusUnidad = unpack("i", Read[50:54])[0]
print("Número de clusters (unidad):", ClusUnidad, "\n")



def ImprimirMenu(): #Función que imprime el menu
    Salir = False
    while not Salir:
        #os.system("clear")
        print(" 1- Visualizar todos los archivos\n","2- Exportar un archivo\n", "3- Eliminar un archivo\n", "4- Importar un archivo\n", "5- Salir\n")
        Opc = int(input("Escriba una opción: "))
        if Opc == 1:
            ImprArchivos()
        elif Opc == 2:
            ExportarArchivo()
        elif Opc == 3:
            EliminarArchivos()
        elif Opc == 4:
            ImportarArchivo()
        elif Opc == 5:
            Salir = True
        else:
            print("Escoja una opción valida\n")
```

Para iniciar el programa primeramente se abre nuestro disquete fiunamfs en modo de lectura y escritura binaria, una vez hecho esto se leerán los contenidos de nuestro superbloque que se encuentran desde 0 – 57, aunque el superbloque abarca los primeros 64 bytes se omiten estos ya que no cuentan con información o elementos necesarios.

Una vez hecho lo anterior se usará “seek” el cual funcionará como un apuntador que se moverá a lo largo de los bits de nuestra unidad. Aseguramos que el apuntador esta en 0 y empezamos con el despliegue de información. Leyendo directo de la imagen leemos según los rangos de bytes en los que se encuentra cada uno de los datos decodificándolos de latín_1. El cambio será en los enteros donde al estar escritos en notación Little endian estos no pueden ser decodificados como con las cadenas anteriores, asi que utilizando unpack se obtendrá una lista con los valores enteros de los bits leídos, pero para no almacenar toda la lista simplemente tomamos el elemento 0 el cual es nuestro número decodificado.

Posterior a esto se muestra la función Imprimir Menu el cual desplegara un menú básico de consola, esta función es llamada justo después de desplegar la información del superbloque.

Listado de contenido.

```
def ImprArchivos():
    #Se llama la función CargaArchivos la cual nos regresara en una lista con los datos de todos los archivos almacenados en el disco
    #Y posteriormente por medio de un ciclo imprimir todos

    Archivos = CargarArchivos()
    print("Listado de archivos:\n", "/    Nombre    / Peso /      Fecha y Hora de modificación      /")
    for elemento in Archivos:
        print("\n",elemento[0]," ", elemento[1]," ", elemento[3])
    print("\n")
```

Para esta primera funcionalidad primero se llamara a la función CargarArchivos()

```
#Función para guardar la información de los archivos almacenados en el sistema
def CargarArchivos():
    Archivos = []
    dire = Size
    nEspacio = 64
    Fecha = None
    #Ciclo para recorrer todos los registros guardados en el disco
    for i in range (nEspacio):
        Imagen.seek(dire)
        dire = dire + 64
        ReadImg = Imagen.read(64)
        #Si en el registro que se esta verificando hay un nombre en vez de "--------------" entonces se guardara
        #el registro con toda su informació en una lista que se usara durante algunas operaciones
        if ReadImg[1:15].decode("latin_1") != "--------------":
            list0 = []#lista temporal
            list0.append(ReadImg[1:15].decode("latin_1"))#Nombre
            list0.append(unpack("i", ReadImg[16:20])[0])#Peso
            list0.append(unpack("i", ReadImg[20:24])[0])#Cluster Inicial
            list0.append(datetime.strptime(ReadImg[38:52].decode("latin_1"),'%Y%m%d%H%M%S'))#Fecha y hora de modificación
            Archivos.append(list0)
    return Archivos
```

Esta función hará un recorrido por el área en que se encuentra la información (no contenido) de cada archivo almacenado en el sistema, para esto por medio de un bucle y mediante intervalos de 64 bits pasara por todos los elementos y en caso que llegado a uno este en la zona del nombre sea diferente a los guiones significara que ese es el registro de un archivo por lo cual se almacenara en una lista temporal y luego se guardara todo este registro en otra lista que será la que se devolverá a nuestra función imprArchivos, claro después de haber pasado por todo el bucle.

Regresando a ImprArchivos una vez se obtiene el listado de archivos simplemente se imprime desplegando al usuario el nombre, peso y fecha de modificación de cada archivo en la unidad.

Exportar archivos.

```
def ExportarArchivo():
    #Se le pedira al usuario el nombre del archivo a exportar y este se mandara a una función donde se checaran todos
    #los elementos guardados viendo si alguno coinvvide con el nombre dado, en caso de encontrar una coincidencia se
    #obtendran los datos del archivo
    NombreArchivo = input("Ingresa el nombre del archivo a exportar(Con formato): ")
    ArchivoExport = BuscarArchivo(NombreArchivo)
    #En caso de que lo devuelto no sea "none" se entrara a la condicional
    if ArchivoExport:
        #Se crea un archivo con el mismo nombre del archivo a exportar y se posicióna el apuntador del disco en la dirección donde se
        #encuentra (clusterinicial * tamañodecadacluster)
        ArchivoNuevo = open(NombreArchivo,"wb")
        Imagen.seek(ArchivoExport[2]*Size)
        #Se guarda en el archivo nuevo los datos almacenados desde la posición del apuntador en el rango de la memoria que ocupa
        ArchivoNuevo.write(Imagen.read(ArchivoExport[1]))
        ArchivoNuevo.close()
        print("Copia realizada\n")
```

Para exportar un archivo primeramente se pedirá el nombre de un archivo, esto será necesario para mandarlo a una función que busque la existencia del archivo en nuestra unidad.

```
def BuscarArchivo(NombreArchivo):
    #Se cargan los archivos existentes
    Archivos = CargarArchivos()
    #Se recorren todos los archivos y se comparan con el Nombre que se manda a la función
    for nFile in Archivos:
        #En caso de encontrarse alguna coincidencia se regresa el registro con la información de ese archivo, en caso con
        if NombreArchivo == nFile[0].lstrip(" "): #lstrip es necesario para quitar los espacios en blanco a la izquierda
            return nFile
    print("Archivo no encontrado\n")
    return None
```

BuscarArchivo primeramente utilizando un listado de todos los archivos obtenido por CargarArchivos hará un recorrido por la lista inspeccionando si alguno de los elementos en el área 0 (Los nombres) coincide con el nombre del archivo indicado por el usuario. Aquí se utiliza lstrip ya que los nombres al contar con espacios sobrantes del lado izquierdo de la cadena nunca habría coincidencias, asi que estos se eliminan.

En caso de encontrar una coincidencia se regresará todo ese registro, de lo contrario se mandará un None.

Pasando nuevamente a la función principal, se verificará que en la variable donde se debería encontrar el registro con los datos del archivo a exportar no sea None. En caso de no ser vacío entonces se creará un archivo que se guardará en nuestra computadora. Para esto primero se establecerá el apuntador en el byte inicial del archivo (Obtenido multiplicando el cluster por el tamaño de cluster) y luego se escribirá en el archivo creado toda la información desde el apuntador en el rango de los bytes que pesa el archivo.

Eliminar archivos.

```
def EliminarArchivos():
    #Al igual que en la función anterior se pide el archivo al usuario y se verifica que ese archivo se encuentre en el disco
    NombreArchivo = input("Ingresa el nombre del archivo a borrar(Con formato): ")
    ArchivoExport = BuscarArchivo(NombreArchivo)
    dire=Size #Posición inicial al terminar el superbloque donde inicia el registro de archivos
    nEspacio = 64 #nEspacio = 64 ya que es el espacio que ocupa el registro de cada archivo
    #Si ArchivoExport no es none se entra a la condicional
    if ArchivoExport:
        #Ciclo que recorrera el disco hasta encontrar un registro que coincida con el nombre obtenido
        for i in range(nEspacio):
            #Se coloca el apuntador en la ultima posición leida y se leen los datos a partir de ahí hasta 64 para
            #obtener todos los datos de un registro
            Imagen.seek(dire)
            Readt = Imagen.read(64)
            #En caso de que lo leido de nuestro disco coincida con el nombre se entrara al ciclo donde se cambiara el nombre del registro
            #a "----------" lo cual quitara el acceso a los datos del archivo
            if Readt[1:15].decode("latin_1").lstrip(" ") == NombreArchivo:
                Imagen.seek(dire+1)
                Imagen.write("----------".encode("latin_1"))
                print("Archivo eliminado\n")
                return
            dire = dire + 64
```

Para esta función se hace un proceso parecido al de exportar un archivo, se pide un nombre de archivo al usuario, se verifica su existencia y si este se encuentra dentro de la memoria entonces se obtendrán todos sus datos.

Entonces en el caso de que este archivo exista se entrará a un bucle parecido al de CargarArchivos donde se recorrerá por intervalos de 64 bits el área de registros de archivos buscando de un nombre igual al que proporciono el usuario. Una vez este sea encontrado se escribirá en esa área del disco “----------“, que si bien esto no elimina el contenido del archivo al indicar que no hay un nombre entonces no se podrá acceder a ese registro dejándolo libre para poder sobrescribirlo.

Importar archivos.

```
def ImportarArchivo():
    #Se pide el nombre de un archivo y se verifica su existencia
    NombreArchivo = input("Ingrese el nombre del archivo a copiar en FiUNAM(Con formato): ")
    ArchivoImport = BuscarArchivo(NombreArchivo)
    ClusterFinal = 0 #Se inicializa una variable para saber el cluster final de los archivos
    dire = Size
    #En caso de que el archivo no exista (ArchivoImport=None) se entra al ciclo
    if not ArchivoImport:
        #Se abre el archivo que se quiere importar y se calcula el tamaño que ocupa
        print("Se puede importar\n")
        ArchivoAImportar =  open(NombreArchivo, 'rb')
        TamañoImportar = os.stat(NombreArchivo).st_size
        #Se obtienen los registros almacenados en el archivo y se ordenan de menor a mayor cluster
        Archivos = CargarArchivos()
        ArchivosOrdenados = sorted(Archivos, key=OrdenarArchivos)
        #Ciclo para buscar si hay espació disponible de almacenamiento entre archivos ya creados
        for x in range(len(ArchivosOrdenados)-1):
            #Se calcula cual seria el cluster final del archivo x
            ClusterFinalA = (ArchivosOrdenados[x][1])/Size + ArchivosOrdenados[x][2]
            #Se obtiene el cluster inicial del archivo siguiente al x
            ClusterInicialB = (ArchivosOrdenados[x + 1][2])
            #Si el tamaño de clusters que ocupa el archivo es menor a los clusters entre el cluster final del archivo x y el inicial de x+1
            #se entra a la condiconal donde se guardara el cluster donde ahora debera ser guardado el contenido del archivo a importar
            if (TamañoImportar/Size) < (ClusterInicialB-ClusterFinalA):
                ClusterFinal = int(ClusterFinalA)
                break
            #En caso contrario el cluster inicial de guardado sera el cluster final + 1 del ultimo archivo
            else:
                ClusterFinal = int((ArchivosOrdenados[x + 1][1])/Size + ArchivosOrdenados[x + 1][2])

        #Ciclo para actualizar el registro de los archivos y guardar su información
        for i in range(Size):
            #Se coloca el apuntador en el inicio de los registros de archivos y se busca donde hay un lugar libre ("---------------")
            Imagen.seek(dire)
            tRead = Imagen.read(64)
            if tRead[0:15].decode("latin_1") == "---------------":
                #Se avanza en uno el apuntador para no modificar el guion del tipo de archivo y despues se almacena la info del documento guardado
                Imagen.seek(dire+1)
                Imagen.write(NombreArchivo.rjust(14).encode("latin_1")+'\x00'.encode("latin_1")+pack("<i", TamañoImportar)+pack("<i", ClusterFinal + 1
                print("Archivo importado\n")
                return
            dire = dire + 64

        #Se posiciona el apuntador del disco en el cluster inicial encontrado en el ciclo anterior y se guarda el contenido del archivo en la memoria
        Imagen.seek((ClusterFinal + 1)*Size)
        ReadImportar = ArchivoAImportar.read()
        Imagen.write(ReadImportar)
        ArchivoAImportar.close()

    else:
        print("Un archivo con ese nombre ya existe en el disco")
```

Aquí nuevamente se pedirá un nombre de archivo al usuario y se buscara su existencia en el directorio, pero ahora con un propósito contrario las veces anteriores ya que se buscara verificar que el archivo no este en el disco, asi que en el caso de que nuestra variable sea none se entrará a la zona para importar.

Primeramente, se abre el archivo a guardar en modo de lectura binaria para un guardado más sencillo además de también de obtener su peso, esto con el propósito de averiguar si cabe en clusters vacíos entre los archivos ya almacenados. Se obtiene el listado de registros y se ordenaran según el cluster inicial de cada archivo, para esto se usa función muy sencilla que solo devuelve los índices del cluster inicial.

```
def OrdenarArchivos(Archivo):
    return Archivo[2]
```

Una vez ordenados se entra a un ciclo que buscara el cluster final de un archivo y el cluster inicial de el archivo posterior a este. Para obtener el cluster final se divide el peso entre el tamaño de losclusters y se le suma el cluster inicial obteniendo asi donde termina ese archivo.

Una vez obtenido ambos clusters se comparan con el tamaño de clusters que ocuparía nuestro archivo a importar (Obtenido dividiendo el tamaño del archivo entre el tamaño de los clusters). En caso de encontrar un espacio adecuado se guardará el valor del cluster final del archivo A, de lo contrario se obtendrá el cluster final del archivo con el cluster inicial más grande.

Después de esto usando una función parecida a la de CargarArchivos se buscará ahora al contrario de la función mencionada un registro con nombre "----------" para sobrescribirla con la información de nuestro archivo.

Finalmente usando el ClusterFinal encontrado se procederá a poner el apuntador en la posición en bytes donde debe guardarse el archivo, siempre usando un cluster más adelante al archivo anterior para evitar sobrescribir información. Entonces se coloca el cursor en la posición inicial, leemos los datos de nuestro archivo a importar y se guarda la información.

ENTORNO DE DESARROLLO

- **Lenguaje de desarrollo:** Python 3.8.3
- **Bibliotecas:**
    1. Os
    2. Datetime
    3. Struct
- **Elementos necesarios:** Para este proyecto al solo hacer uso de la consola como interfaz no necesita de elementos extras a Python y las bibliotecas que se incluyen en este. Pero para una ejecución correcta se debe remarcar de la necesidad de tener una imagen de disco con el nombre de fiunamfs. En este caso se utiliza el proporcionado por el profesor. No olvidar que este debe estar en la misma carpeta de la que se ejecute el código.
- **Entorno de trabajo:**
    1. SO: Windows 10 pro versión 21H2
    2. Maquina: Ryzen 3400g con 16 gb de ram

- **Forma de Ejecución y Ejemplos de uso:**
    1. Primeramente, se deberá compilar el programa a lo cual se nos presentará el inicio del sistema con sus características y posteriormente el menú de manejo del programa.
    2. Una vez se presenta el menú se le pedirá al usuario que escoja una de las opciones a realizar, para esto se deberá escribir el numero a la izquierda de las acciones mostradas según lo que se busque y posteriormente se dará enter.

```
          (Micro)Sistema de archivos: FiUnamFS
Versión:    23.1

Etiqueta volumen:    FI-UNAM 2023-1

Tamaño cluster: 1024 bytes

Numero de clusters (directorio): 4

Número de clusters (unidad): 720

 1- Visualizar todos los archivos
 2- Exportar un archivo
 3- Eliminar un archivo
 4- Importar un archivo
 5- Salir


Escriba una opción: 1
```

    3. Después de escoger nuestra opción tendremos 5 posibilidades

Opción 1 (Visualizar todos los archivos)

```
Listado de archivos:
 /    Nombre    / Peso /      Fecha y Hora de modificación      /

     README.org   31078   2022-12-08 17:19:11

     Prueba.txt   10   2023-01-05 21:25:58

       logo.png   188055   2022-12-08 17:19:11

   mensajes.png   296326   2022-12-08 17:19:11
```

Si elige la opción 1 se hará un despliegue de todos los archivos guardados en el sistema, además de mostrar el peso en bytes de cada uno y la Fecha y hora de su última modificación. Posterior a esto se volverá a desplegar el menú.

Opción 2 (Exportar un archivo)

Al escoger la segunda opción se exportará un archivo que se contenga en la unidad, por lo cual primero se preguntará al usuario que archivo desea exportar.

```
Ingresa el nombre del archivo a exportar(Con formato): logo.png
Copia realizada
```

Una vez escribimos y damos enter al nombre escrito nuestro sistema exportara el archivo solicitado en el mismo directorio donde se encuentre el código.

fiunamfs	05/01/2023 09:30 p. m.	Archivo de image...	720 KB
logo	05/01/2023 10:21 p. m.	Archivo PNG	184 KB
proyecto	05/01/2023 10:21 p. m.	Python File	10 KB

¿Qué había en los archivos cargados por el profesor?

¡Buen trabajo!

```
#+title: Proyecto 2: (Micro) sistema de archivos

#+BEGIN_SRC yaml
Planteamiento: 2022.12.01
Entrega: 2023.01.05
#+END_SRC

# ¡Las [[./calificaciones.org][calificaciones y comentarios]] ya están disponibles!

** Descripción del proyecto

Para la unidad de sistemas de archivos, creo que resulta natural que
el proyecto sea implementar un sistema de archivos 😉 Para esto, lo
harán trabajando sobre una /especificación/ y sobre un /caso de
referencia/.

** ¿Qué tengo que hacer?

Lo que ustedes deben desarrollar es un programa que pueda obtener,
crear y modificar información en el micro-sistema-de-archivos de la
Facultad de Ingeniería, =FiUnamFS=.
```

Opción 3 (Eliminar un archivo)

Para esta tercera opción al igual que en la anterior se deberá ingresar el nombre de un archivo, pero ahora este debe encontrarse dentro del sistema de archivos. Una vez escogemos el archivo este procederá a eliminarse.

```
Escriba una opción: 3

Ingresa el nombre del archivo a borrar(Con formato): Prueba.txt
Archivo eliminado
```

```
Listado de archivos:
 /    Nombre    / Peso /      Fecha y Hora de modificación      /

     README.org   31078   2022-12-08 17:19:11

       logo.png   188055   2022-12-08 17:19:11

   mensajes.png   296326   2022-12-08 17:19:11
```

Opción 4 (Importar un archivo)

En la cuarta opción se guardará un elemento desde nuestra computadora a el sistema de archivos, este archivo debe encontrarse en la misma carpeta donde se ejecuta el código. Como en las opciones anteriores se pedirá el nombre del archivo a importar para posteriormente guardarse en nuestro sistema.

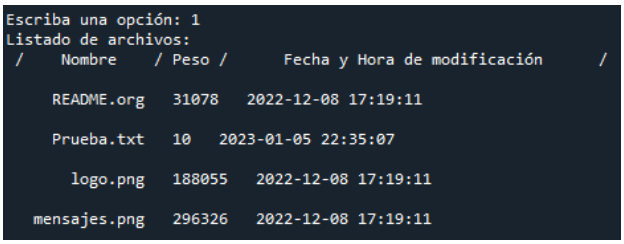
```
Escriba una opción: 4

Ingrese el nombre del archivo a copiar en FiUNAM(Con formato):
Prueba.txt
Archivo no encontrado

Se puede importar

Archivo importado

 1- Visualizar todos los archivos
 2- Exportar un archivo
 3- Eliminar un archivo
 4- Importar un archivo
 5- Salir
```



Opción 5 (Salir)

Esta ultima opción simplemente terminara con la ejecución del programa.