Proyecto 3

Sistemas Operativos

Carolina Mota García **24/11/2023**

La problemática:
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.

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

Listar los contenidos del directorio
Copiar uno de los archivos de dentro del FiUnamFS hacia tu sistema
Copiar un archivo de tu computadora hacia tu FiUnamFS
Eliminar un archivo del FiUnamFS
Desafortunadamente, este sistema de archivos simplote es muy dado a la fragmentación externa. Generen también un programa que desfragmente al FiUnamFS. Ojo, la defragmentación debe hacerse dentro del sistema de archivos (no creando un sistema de archivos nuevo y copiando hacia éste).

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, en todas las estructuras de FIUnamFS, las cadenas de texto deben ser ASCII 8-bit (no Unicode UTF-8).
En las estructuras del disco, todos los enteros serán representados como valores de 32 bits, en formato little endian.
Esto es, por ejemplo: El valor 1354 se representará como los bytes (74, 5, 0, 0) (porque ((0 * 256 + 0) * 256 + 5) * 256 +
74 = 1354). El valor 1234567890 se representará como los bytes 210, 2, 150, 7 (porque ((7 * 256 + 150) * 256 + 2) * 256 + 210
= 1234567890.
Para hacer las conversiones desde o hacia este formato, les sugiero usar el formato <I con las funciones pack() y unpack(), disponibles en distintos lenguajes (Python: struct.pack(), Ruby: Array#pack(), Perl: pack(), PHP: pack(), hay varias biblitecas que pueden usar para Javascript, como node-jspack o bufferpack). Si van a usar C o C++, les sugiero revisar las funciones htons() y ntohs(). ¿Java? Si bien no forman parte del lenguaje, pueden encontrar varias implementaciones de htons() y ntohs() en la red.
La superficie del disco se divide en sectores de 256 bytes. Cada cluster mide cuatro sectores.
El pseudodispositivo 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 pseudodispositivo es el superbloque. 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.
10–14
Versión de la implementación. Estamos implementando la versión 24.1 (como cadena, no como número). Deben validar que el sistema de archivos a utilizar sea de esta versión, para evitar la corrupción de datos.
20–39
Etiqueta del volumen
40–44
Tamaño del cluster en bytes
45–49
Número de clusters que mide el directorio
50–54
Número de clusters que mide la unidad completa
El resto del superbloque puede quedar vacío (o puedes sobrecargarlo con otros datos — No es importante para la implementación.

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
Tipo de archivo. Dado que el sistema de archivos actual no tien soporte para directorios, dispositivos, pipes, ni otros archivos especiales, todos los archivos tendrán el caracter - (0x2d, 45). Las entradas vacías tendrán el caracter / (‘0x0f’, ‘47’)
1–15
Nombre del archivo
16–19
Tamaño del archivo, en bytes
20–23
Cluster inicial
24–38
Hora y fecha de creación del archivo, especificando AAAAMMDDHHMMSS (p.ej. ‘20231108182600’ para 2023-11-08 18:26:00)
38-52
Hora y fecha de última modificación del archivo, especificando AAAAMMDDHHMMSS (p.ej. ‘20231109182600’)
52–64
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.

Explicación de resolución:

Lenguaje utilizado:
Python 3.0

Bibliotecas usadas:
struct
math

El problema se divide en 5 puntos principales. La lectura del directorio, copiar un archivo, pegar un archivo, borrar un archivo y desfragmentar.

Inicialmente, necesitamos definir algunas variables.

Definimos el archivo a evaluar

```
mainFile = "fiunamfs.img"
img = open(mainFile,"br+")
```

Definimos las variables correspondientes a los clusters y al tamaño del directorio, así como la lista de los archivos y el almacenamiento del mapa.

```
sizeCluster= unpackData(40,4)
dirClusters = unpackData(45,4)
UnClusters = unpackData(50,4)
dirSize = 64
fileList=[]
storageMap = []
```

Para lograr cualquiera de esas acciones, es importante inicialmente desarrollar una función que permita traducir la información dentro de FIUnamFS hacia una con la que podamos trabajar.

Es por ello que utilizamos initMap para inicializar el mapa.

```
#Inicializa el mapeo
def initMap():
    global storageMap
    for x in range(5):
        storageMap.append(1)
    while (len(storageMap) != 720):
        storageMap.append(0)
```

Por otra parte, utilizamos initFiles para inicializar los archivos.

```
def initFiles():
    global fileList
    sizeDir = int((sizeCluster * dirClusters)/dirSize)
    for x in range(sizeDir):
        res = getData(x)
        if(res != None):
            fileList.append(res)
            updateMap()
```

Y de esta manera, tenemos nuestro directorio base listo para ser utilizado y comenzar a modificarlo o revisarlo.

Para el primer punto, necesitamos enlistar los elementos dentro del directorio, por lo que por cada elemento (file) que exista, se imprimirá el nombre y tamaño del archivo.

```
def showDir():
    global fileList
    for x in fileList:
        print(str(x.name)+"        "+str(x.size)+" bytes")
```

Sin embargo, no se imprimirán los archivos vacíos o con tamaño de byte 0, debido a que getData, dentro de initFiles, omite los archivos con esas características.

Prueba:

```
Bienvenidx a FIUnamFS

Opciones:
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) Desfragmentar
6) Salir

1
El contenido del directorio es

README.org        31209 bytes
logo.png        170192 bytes
mensaje.jpg        102657 bytes

Opciones:
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) Desfragmentar
6) Salir
```