Sistemas Operativos

(Micro) sistema de archivos

Proyecto #4

Barrero Olguín Patricio
Espino Rojas Héctor Daniel.

Objetivo:

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.

Requerimientos:

- 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 14400 Kilobytes (les había dicho *1440*, pero me sobró un cero en mi programa de creación)
- 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 clúster 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.

Descripción

Este proyecto fue desarrollado en Windows 10 y Debian Buster. Se utilizan abstracciones que permiten manejar los archivos en distintas plataformas. Presentamos una interfaz mediante línea de comandos, generando nuestro propio interprete el cual soporta las operaciones mencionadas por el profesor. Y realiza las respectivas validaciones sobre los comandos.

El sistema de archivos trabaja tomando un archivo que representa el diskette, para cada operación de I/O a los distintos ficheros se simula los accesos a disco abriendo y cerrando el archivo. De este modo cerrando con las operaciones de lectura y escritura, porque de otro modo Python los mantendría en un estado inseguro. El abrir el interprete es equivalente a montar el sistema de archivos. Se utiliza un primer ajuste para seleccionar donde se encontrará el archivo.

La definición de alto nivel de los comandos permite tener una la interfaz – interprete separado de la implementación detallada. Dicha implementación a "bajo nivel" es la encargada de trabajar con el archivo y se sustenta funcionalmente en realizar operaciones de lectura y escritura. Estas operaciones son realizadas mediante manejo del apuntador seek y el calculo de direcciones. Por ejemplo, calculando el cluster inicial multiplicado por el tamaño del cluster, obtendremos el tamaño del archivo.

Importante hacer notar que se incluyó una funcionalidad de mostrar el contenido de un archivo mediante un comando llamado cat. En la opción de borrar archivo se siguió un modelado similar al de FAT,

ya que la entrada en el superbloque es marcada como libre y el archivo en si mismo no es eliminado, sino que sus datos son vistos ahora como espacio libre y eventualmente podrían ser sobrescritos.

El proyecto consta de los siguientes archivos:

main.py -> Define la interfaz gráfica, realiza validaciones de los datos iniciales y manda a llamar a los demás módulos.

sistema_archivos.py -> Contiene las clases: archivo y sistema de archivos.

archivo -> Parsea de una cadena del archivo exadecimal a datos manejables.

sistema de archivos -> Implemantacion de los comandos

trabajar_archivos -> Define las funciones para trabajar los archivos.

tratamiento_cadenas -> Realiza las codificaciones y algoritmos requeridos por el programa.

fecha.pý -> Define funciones para manejo de cadenas de fechas.

Requerimientos:

*Python =3.6.9
--El resto de modulos utilizados se encuentran dentro del core de Python 3.6.9
--Por lo tanto, no es necesario instalarlos

Módulo:
colorama. = 0.4.3
math =
Instalación de los módulos externos (para python3 en linux/mac os):

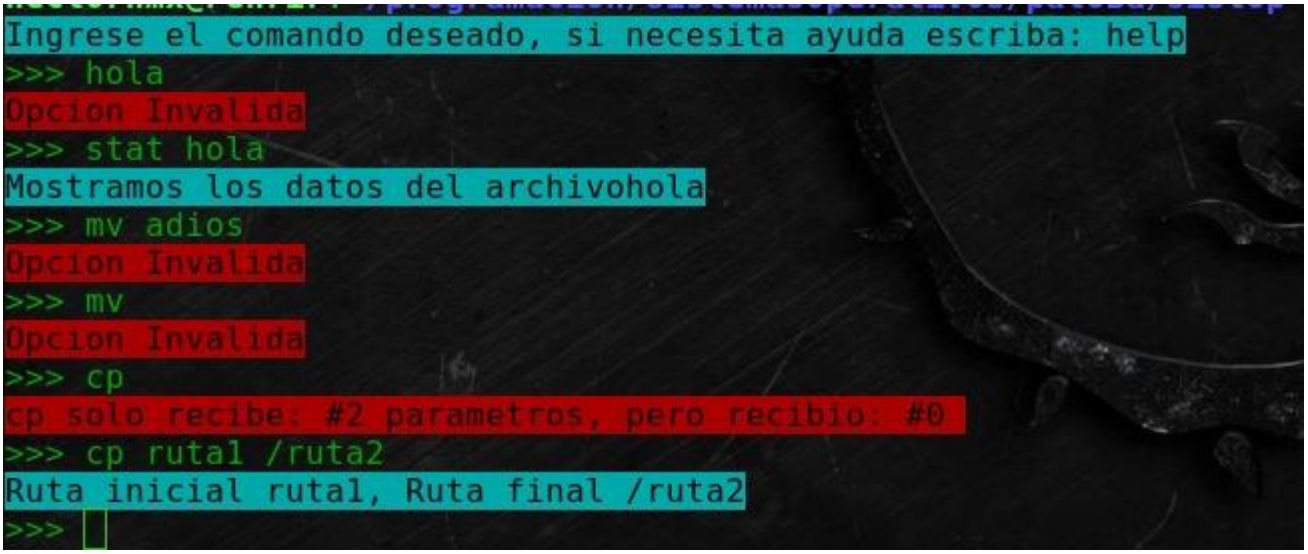
pip3 install colorama

pip3 install math

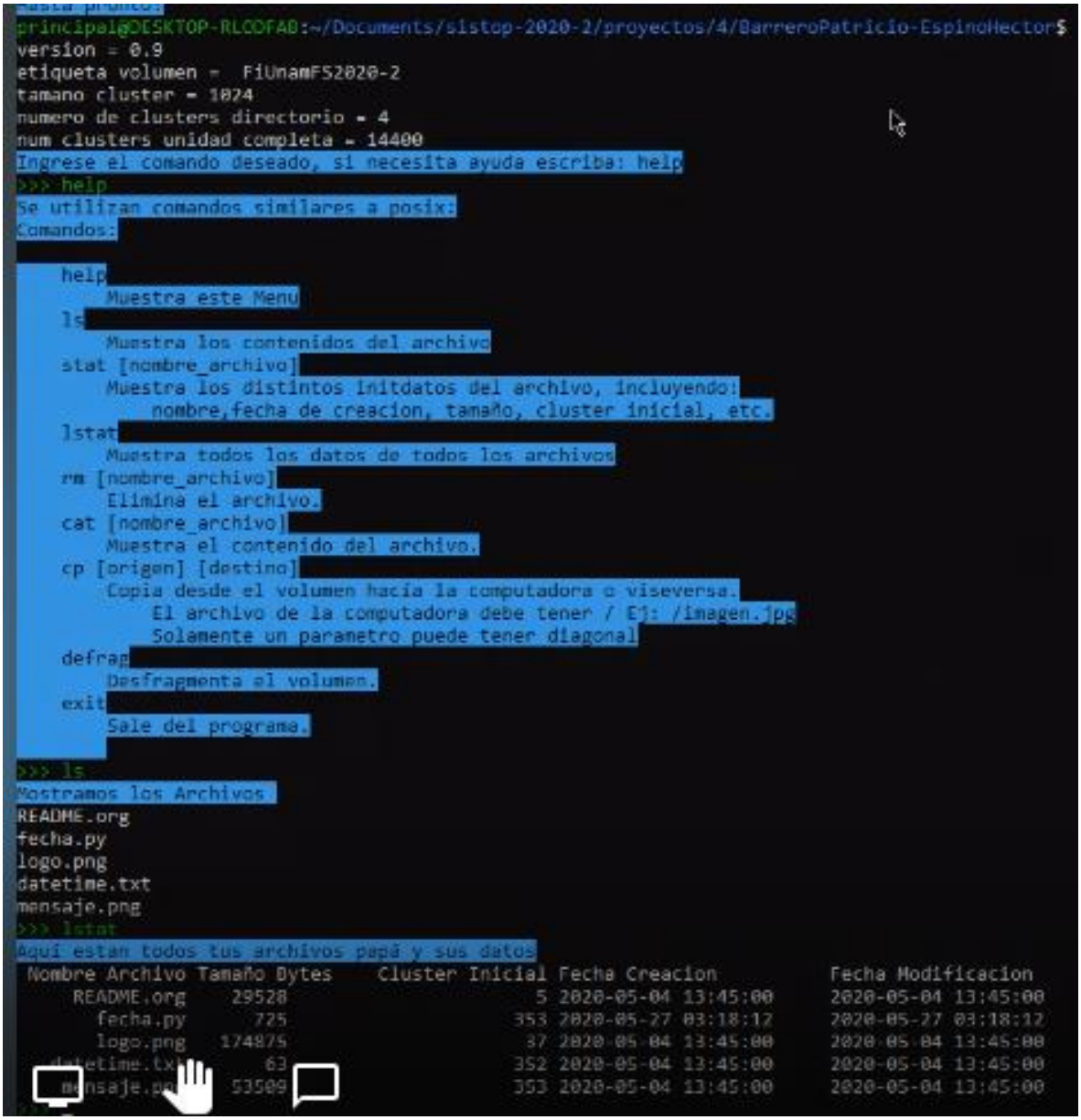
Ejecución
linux/mac os:
python3 main.py
windows:
python main.py

Uso

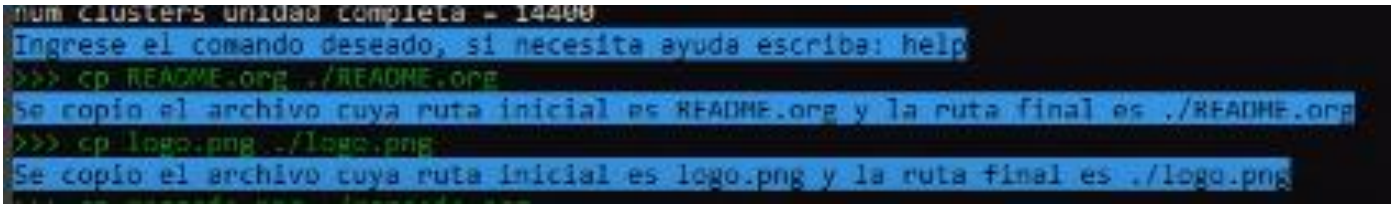
Al iniciar el programa se nos será mostrado un prompt donde podremos ingresar los comandos deseados. En caso de requerir conocer la lista de comandos escribimos help. El programa sigue un coidog de colores el cual nos permite conocer cuando una operación fue llevada a cabo de forma exitosa. Enfocándose el interprete en dar claras intenciones de los resultados en lugar de un enfoque estético.



Pruebas de funcionalidad:



Prueba copiando del FS a la computadora readme:



Y obtenemos