Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Sistemas Operativos

Proyecto 3 -File System

Profesora:

Erika Marín Schumann

Estudiantes:

María José Cortés - 2018138674

Esteban Cruz López - 2018104794

Luis Venegas Leiva – 2019079322

I Semestre 2022

Contenido

Introducción

Este proyecto se realizó como parte del curso de Principios de Sistemas Operativos de la carrera de Ingeniería en Computación en el Instituto Tecnológico de Costa Rica en el primer semestre de 2022 con la profesora Erika Marín Shumann.

Se trata de un administrador de archivos que mediante comandos y demás funcionalidades permite realizar operaciones como creación, modificación, copias y eliminación de archivos, además de mostrarse la estructura de este mediante una interfaz de usuario, que también permitirá realizar algunas operaciones como abrir, ver propiedades y eliminar archivos y directorios mediante opciones desplegadas con click derecho.

Además, se debe realizar un archivo que simule el disco del file System que se está creando.

El proyecto será desarrollado en el lenguaje de programación Python y la interfaz se diseñará con la ayuda de la librería tkinter.

Estrategia de solución

El proyecto fue desarrollado en el lenguaje de programación Python bajo una estructura MVC (Modelo, Vista, Controlador), por lo que para la interfaz gráfica se utilizó la librería tkinter, la cual muestra la estructura del File System y una consola en la que se ingresan los comandos, para procesar estos comandos se utilizó el patrón de diseño “Command”, que permite procesar cada funcionalidad según el comando ingresado, para la estructura general del proyecto se crearon tres clases, Elemento, Directorio y Archivo, la clase Elemento es la clase padre que contiene atributos como id y nombre, la clase Directorio es el pilar del

FileSystem, ya que contiene listas tanto de Archivos como de Directorios, lo que permite que se puedan realizar subdirectorios tanto como sea necesario.

Análisis de Resultados:

Se cumplió a cabalidad con las funcionalidades solicitadas logrando implementar cada uno de los comandos requeridos. A excepción de la funcionalidad de copiar que está a un 50% porque faltó la copia de directorios.

Además, se cumplió con los patrones de diseño planteados logrando que la solución sea comprensible y simple.

Funciones implementadas y porcentaje de completitud:

1. Inicializar FileSystem: 100%
2. Crear un archivo: 100%
3. Crear un directorio: 100%
4. Cambiar de directorio de trabajo:
5. Listar directorios: 100%
6. Modificar archivo: 100%
7. Ver propiedades: 100%
8. Ver contenido: 100%
9. Copy: 50% Los tres modos funcionan para archivos pero no para directorios
    a. Un elemento externo dentro de nuestro sistema de archivos.
    b. Un elemento interno a una ruta externa.
    c. Un elemento virtual hacia una ruta virtual.
10. Mover: 100%
11. Eliminar: 100%
12. Find: 90% Justificación: no se implementó regex para encontrar todos los archivos con nombre que coincida con un patrón
13. Tree: 100%

Lecciones aprendidas

Durante el desarrollo de la solución de este proyecto conseguimos aprender valiosísimas lecciones las cuales se presentan a continuación:

- Aprendimos sobre el manejo de clases abstractas e interfaces en Python, así como mejoramos nuestro conocimiento en la aplicación de patrones de diseño como el MVC y el Command.
- Aprendimos bastante en la administración del espacio de la memoria secundaria, particularmente en la asignación enlazada de sectores bajo el método FIRST FIT.
-

Pruebas

Como parte de la solución se incluyeron 3 pruebas en las que, si se ejecutan se valida, al menos visualmente, el funcionamiento de nuestra aplicación. Estas pruebas se encuentran definidas dentro del archivo “Pruebas.py” y se ejecutan por medio de la llamada a alguna de las pruebas, por defecto se ejecuta la prueba 1 la cual carga un ejemplo de sistema de archivos y ejecuta varias llamadas a comandos.

A continuación, se presentan screenshots del código y resultado de ejecución para cada una de las pruebas:

Prueba1

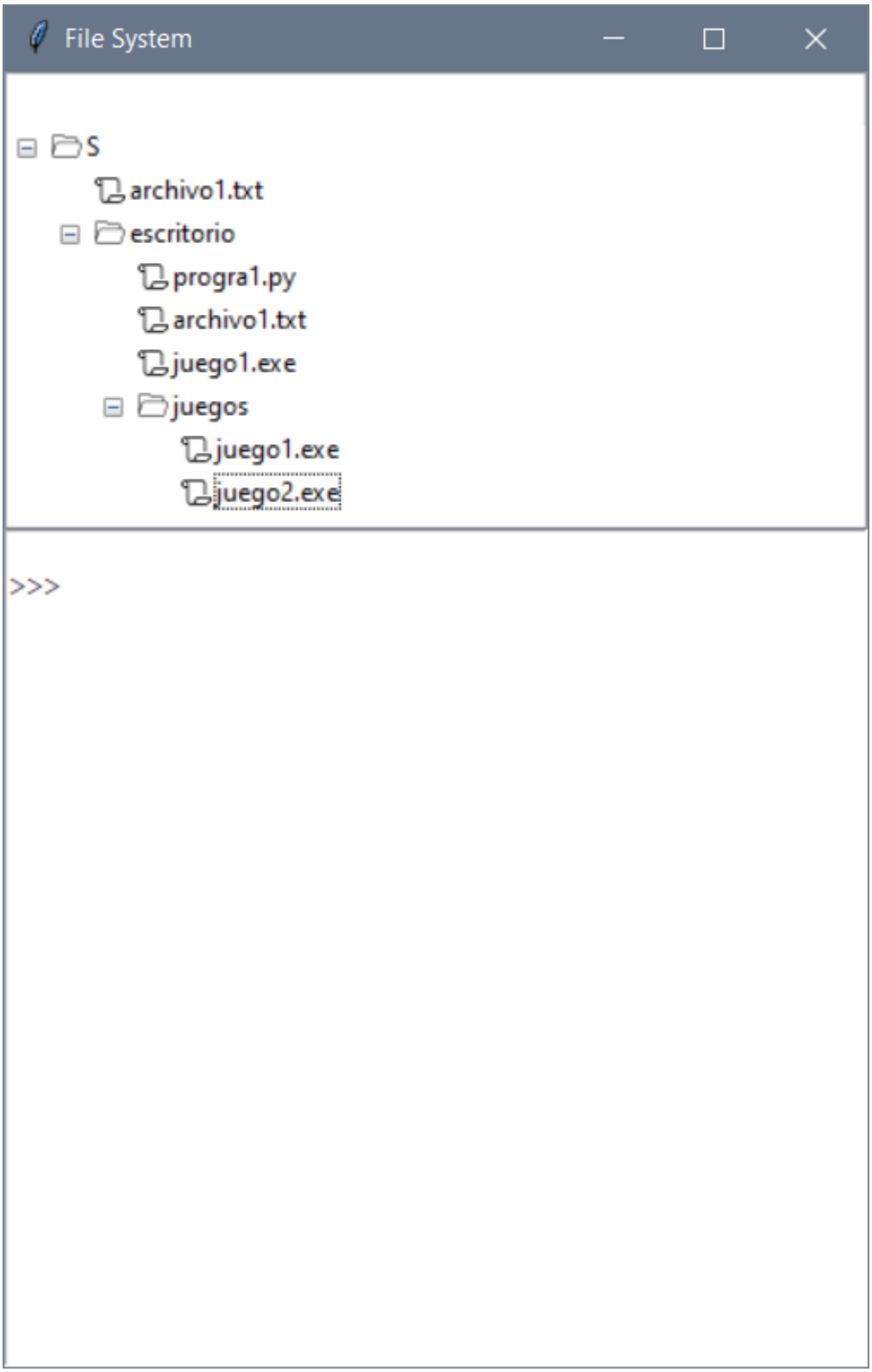
Código:

```
def prueba1():
    import Vista
    import FileSystem

    vista = Vista.Vista()


    FileSystem = vista.FileSystem
    FileSystem.inicializar("S", 16, 8)
    FileSystem.crear_archivo("archivo1.txt", "contenido1")
    FileSystem.crear_directorio("escritorio")
    FileSystem.cambiar_directorio("escritorio")
    FileSystem.crear_directorio("juegos")
    FileSystem.crear_archivo("progra1.py","print('hola mundo')")
    FileSystem.cambiar_directorio("juegos")
    FileSystem.crear_archivo("juego1.exe", "contenido2")
    FileSystem.crear_archivo("juego2.exe", "contenido3")
    FileSystem.crear_archivo("juego3.exe", "cont")
    FileSystem.modificar_archivo("juego3.exe", "Este es el juego 3")
    FileSystem.copiar("S/archivo1.txt", "S/escritorio","-v")
    FileSystem.copiar("juego1.exe", "Archivos","-vl")
    FileSystem.borrar_archivo("juego3.exe")
    FileSystem.copiar("Archivos/juego1.exe", "S/escritorio","-lv")
    vista.actualizarArbol()
    vista.mainloop()
```

Resultado:



El disco:

contenido1 print('hola mundo') contenido2 contenido3 contenido2 contenido1

Prueba2:

Código:

```
def prueba2():
    from DiskManager import DiskManager
    disco = DiskManager('1', 4, 4)
    disco.escribir("as.txt","aaaa")
    disco.escribir("bs.txt","bb")
    assert disco.escribir("cs.txt","ccccccccc") == -1
    disco.eliminar("as.txt")
    disco.escribir("cs.txt","ccccccccc")
    with open("Discos/Disco 1.txt", "r") as f:
        assert f.readline() == 'ccccbb  cccccc  '
```

Resultado:

Finaliza con éxito, sin errores.

Prueba 3:

Código:

```
def prueba3():
    from DiskManager import DiskManager
    disco = DiskManager('2', 4, 4)
    disco.escribir("a.txt","aaaa")
    disco.escribir("b.txt","bb")
    disco.escribir("a.txt","ccccccccc")
    with open("Discos/Disco 2.txt", "r") as f:
        assert f.readline() == 'ccccbb  cccccc  '
```

Resultado: finaliza con éxito, sin errores.

Manual de usuario

Para la ejecución de nuestra solución, se debe utilizar la versión 3.8.3 de Python.

Para abrir el programa se debe ejecutar el archivo pruebas.py

Bitácora de trabajo

Miércoles 1 de junio: Reunión inicial de todos los miembros del grupo para ponernos al punto adecuado para empezar a trabajar. Duración: 2 horas.

Del miércoles 1 de junio a miércoles 8 de junio: Se trabajó de manera asincrónica en lo especificado en la reunión anterior.

Del jueves 9 de junio al martes 14 de junio: Se terminaron detalles y se comenzaron a hacer las pruebas respectivas al programa.

Del miércoles 15 al viernes 18: Se redactó la documentación completa, con colaboración de los tres miembros del equipo, por medio reuniones virtuales.

Bibliografía

https://stackoverflow.com/questions/31414263/python-using-open-w-filenotfounderror

https://stackoverflow.com/questions/7165749/open-file-in-a-relative-location-in-python