

Universidad de Santiago de Chile
Facultad de Ingeniería
Departamento de Ingeniería Informática

INFORME LABORATORIO 3

(PARADIGMA ORIENTADO A OBJETOS - JAVA)

Alumno: Cristóbal Torres Undurraga
Profesor: Roberto González I.
Fecha: 18 de Julio de 2023
Asignatura: Paradigmas de Programación

1. Introducción

El informe consiste en 11 partes: Introducción, Descripción del Problema, Descripción del Paradigma, Análisis del Problema, Diseño de la Solución, Aspectos de la Implementación, Instrucciones de Uso, Resultados, Conclusiones, Referencias y Anexos.

En el siguiente informe se explicará la implementación de un sistema operativo con el paradigma orientado a objetos, en específico, el lenguaje de programación Java. El paradigma orientado a objetos (POO) corresponde a un tipo de programación que se basa principalmente en objetos, clases y el cómo se relacionan/trabajan entre sí.

2. Descripción del Problema

El problema consiste en la simulación de un sistema operativo (Unix-like) simplificado. Debe tener un sistema de archivos (file system), el cual contiene procesos, métodos y reglas. El sistema operativo debe ser capaz de realizar operaciones a través de comandos, los cuales crean, modifican o eliminan elementos del sistema de archivos. Debe tener 5 partes esenciales: sistema, drivers, usuarios, carpetas y archivos. El desafío está en cómo abstraerse del problema y poder organizar el sistema para que se implementen todas las partes y funcionen entre sí por medio de operaciones (**register**, **login**, **mkdir**, **addFile**, **switchDrive**, etc).

3. Descripción del Paradigma

El paradigma utilizado en esta instancia es el paradigma orientado a objetos (POO), este se basa en la programación utilizando herencia. La abstracción de datos es programar usando tipos de datos definidos por el usuario, con algunas excepciones, el paradigma orientado a objetos puede y aspira a ser un superset de tipos de datos abstractos (Stroustrup, 1988). Este es un paradigma que pertenece a la familia imperativa, familia de paradigmas que basa las soluciones en declaraciones que alteran el programa. El paradigma se basa en el concepto de “Objeto”, con tipos de objetos denominados “Clases”. Cada clase tiene características y comportamientos. Una clase se puede operar con otra mediante relaciones, éstas pueden ser composición, agregación, asociación y dependencia, representando distintos niveles. Las clases también pueden heredarse de otras, permitiendo clases más específicas o con comportamientos distintos, mediante el polimorfismo, que permite realizar la misma acción de forma distinta, ya sea con una sobreescritura (misma firma y aridad pero realiza una acción distinta) o una sobrecarga (mismo identificador, distinta aridad y/o tipo de dato). Existen las interfaces, las cuales permiten a las clases implementar comportamientos bajo un contrato, definiendo el **qué** pero no el **cómo**. (Gacitúa, 2023). El paradigma orientado a objetos tiene muchos conceptos útiles que ayudan a tener una mejor comprensión y abordar el problema de forma más intuitiva.

4. Análisis del Problema

Para crear y generar el sistema operativo y sistema de archivos con el paradigma orientado a objetos se escoge como representación clases implementan interfaces y se relacionan a través de ellas. Las clases Drive, File, Folder y User deben estar relacionadas al sistema de alguna manera, y este último debe conectarse con el Menú para poder realizar las operaciones correspondientes.

Uno de los problemas más mencionados en los requisitos funcionales son la unicidad en el sistema de archivos o directorio, que cada elemento en el sistema o directorio no se repita, métodos como ***addDrive***, ***register***, ***login***, ***switchDrive***, ***mkdir***, ***addFile***, ***copy***, ***move*** y ***ren***, esto se debe resolver buscando en cada directorio o en el sistema si existe otro elemento que tenga un atributo igual (Letra, nombre o ruta).

Otro punto importante es poder agregar elementos al sistema. Cumpliendo la unicidad, y utilizando ArrayList, es posible añadir nuevos elementos a las listas, simplemente usando el método “add”, para agregar el elemento al final de la lista.

El último punto general es poder eliminar carpetas o archivos del sistema, como realiza el método ***del***. En lugar de crear una papelera y estar moviendo archivos y carpetas de un lado a otro, se consideró mejor hacer uso de un atributo “eliminado”, este es un booleano que determina si el elemento ha sido eliminado o no, mostrándose en el apartado de “Papelera” al visualizar el sistema. En el caso de que vacíe la papelera, es cuando el elemento es realmente eliminado del sistema.

5. Diseño de la Solución

Para formar y operar dentro del sistema operativo, se hace una representación mediante clases e interfaces. La clase Filesystem tiene unidades (Drive), usuarios (User), carpetas (Folder) y archivos (File), además de atributos relativos al sistema como el nombre, la fecha de creación, el usuario que ha iniciado sesión, el drive que está fijado y la ruta en que se realizan las operaciones.

La clase Drive tiene una letra que es única que la identifica, un nombre y una capacidad. La clase Folder tiene un nombre, ruta, creador, fecha de creación, fecha de modificación y un atributo booleano que indica si la carpeta ha sido eliminada. La clase File es abstracta, ya que solo se trabaja con sus subclases más específicas, esta tiene un nombre, tipo, extensión, ruta, contenido, fecha de creación y el indicador eliminado. Sus subclases pueden ser de 3 tipos: Texto Plano, Documento y Código Fuente, pero todas se trabajan de la misma forma, ya que heredan los mismos atributos y métodos que la clase padre. La clase User tiene un nombre y fecha de creación. Todos se comunican entre sí mediante las interfaces, no directamente entre las clases. El sistema completo es parte de la clase Menu, la cual solo tiene métodos relativos a realizar cambios en el sistema. [Diagrama 2]

En lugar de que cada archivo se encuentre dentro de una carpeta y que cada carpeta esté dentro de una unidad, se determina donde pertenece cada elemento mediante sus atributos. Para las carpetas, su localización en el sistema depende del primer caracter de su ruta, que define la unidad, y la ruta como tal. Los archivos dependen de su ruta que las contiene. Con esto, se evita tener una representación arbórea del sistema de archivos.

Para realizar operaciones en el sistema se utiliza el Menú, el cual indica con números cuáles son las opciones disponibles.

Para la creación de elementos (**system**, **addDrive**, **register**, **mkdir**, **addFile**) se debe verificar que se cumplan los requisitos de unicidad dentro del sistema o el directorio en que se están realizando las operaciones. Para esto se hace uso de métodos del tipo *buscar*, que determinan mediante iteración si es que un elemento ya existe en el sistema o directorio.

Para la modificación de datos del sistema o de elementos (**login**, **logout**, **switchDrive**, **cd**, **del**, **copy**, **move**, **ren** y **format**), primero se verifica la existencia de los elementos en el sistema mediante métodos *buscar* y después se modifican los atributos que requiera el método, respetando la unicidad. Los métodos **copy** y **move** además deben verificar que la ruta objetivo exista en el sistema. Los métodos **del** y **format** eliminan elementos del sistema, con la diferencia de que **del** solamente cambia el atributo “eliminado” del elemento, en cambio **format** quita todo lo que esté relacionado a la unidad ingresada.

Algunos métodos pueden trabajar tanto en archivos como carpetas (**del**, **copy**, **move** y **ren**). Para determinar cuál fue el elemento ingresado, se comprueba si existe un carácter “.” en el String ingresado y se determina donde se encuentra. Si se encuentra al medio, entonces es un archivo, en cualquier otro caso corresponde a una carpeta. Al trabajar con carpetas, es necesario también modificar los subdirectorios de la carpeta, estos se verifican usando la ruta actual como un substring y después se compara con los demás directorios.

En el caso de **dir**, se forma un string que tiene los contenidos del directorio, esto incluye carpetas y archivos, dependiendo de los parámetros ingresados.

6. Aspectos de la Implementación

La implementación del proyecto se estructuró en 5 TDAs como interfaces implementadas por clases: Filesystem, Drive, Folder, File y User. Con la particularidad de que File es una clase abstracta, que hereda a 3 clases: Texto Plano, Documento y Código Fuente. Cada TDA tiene constructores, selectores, modificadores y otras operaciones, con sus respectivas documentaciones. Además, existe la clase Menu, que contiene el Filesystem y realiza operaciones en el mismo.

Para el desarrollo del laboratorio, se utilizó el sistema operativo Manjaro GNU/Linux con la arquitectura x86_64, se ocupó OpenJDK versión 11, en el IDE IntelliJ IDEA Ultimate 2022.3. Para ejecutar el programa, se usa un archivo Bash.

7. Instrucciones de uso

Para utilizar el programa, es necesario ejecutar el archivo “ejecutar.sh”, Una vez dentro, el programa entrega un menú en el cual se debe crear un sistema. [Figura 1] Una vez creado, se muestra un menú con el que se puede crear otro sistema, modificar el sistema, visualizar o listar directorio. [Figura 2] Si se crea un nuevo sistema, se borra el anterior. El sistema solo permite registrar usuarios y crear unidades, hasta que se fije una unidad con la cual trabajar. [Figura 3] El menú pedirá distintas entradas dependiendo de la operación que se realice. Por ejemplo, agregar un archivo requiere que se introduzca un nombre y contenido. [Figura 4] Los cambios son visibles al visualizar el sistema o al listar el directorio. [Figura 5]. Un caso particular es listar el directorio (**dir**), ya que los parámetros deben ser escritos de la forma “arg1, arg2, arg3”. [Figura 6]

8. Resultados

La mayoría de los requerimientos abordados lograron un alcance completo (a excepción de ciertas funcionalidades de **cd**, **del**, **copy**, y **dir**). [Tabla 1] Se hicieron pruebas con todas las operaciones, para verificar los alcances que, que no sean case-sensitive, no agreguen elementos si estos rompen la unicidad y que se logre trabajar entre distintas unidades y directorios.

Los requerimientos funcionales no completados fueron: **cd**, se implementó la forma base del método incluidos "/", ".." y ".", pero en este último no se realizó una implementación adecuada. **del**, el método logra eliminar tanto archivos como carpetas, pero no puede usar el comodín "*". **copy**, logra operar sobre archivos y carpetas por igual y hace uso del comodín "*", pero solamente si se encuentran en el mismo directorio. **dir**, solo implementaron los parámetros "/o [-]N", "/?" y la versión simple. Todas las implementaciones no realizadas o completas fueron debido a una falta de tiempo, más no por un desconocimiento del paradigma

9. Conclusiones

Al momento de abordar el problema desde un punto de vista imperativo, se denota una gran organización de los elementos, permitiendo acceder a niveles de abstracción mayores de forma simple, mediante el uso de clases.

Uno de los puntos negativos que tiene el paradigma orientado a objetos, son la cantidad de conceptos que conlleva, que a pesar de permitir soluciones claras y detalladas, esto impide que el paradigma sea tan accesible para el resto, tal como lo es el paradigma procedural o en cierta medida el paradigma funcional.

En comparación con los paradigmas declarativos, en este caso Funcional y Lógico, el paradigma orientado a objetos consigue un mayor nivel de orden de los elementos, y un mejor manejo de los datos que los de su contraparte declarativa. Los paradigmas anteriores destacaban por su simplicidad y flexibilidad para trabajar con distintos problemas, principalmente porque se enfocan en el **qué** y no en el **cómo**, en cambio, el paradigma orientado a objetos se destaca por sus capas de abstracción, consiguiendo llegar a soluciones complejas que a la vez son mantenibles en el tiempo. Los resultados obtenidos entre los distintos paradigmas fueron muy cercanos, con la diferencia en el proceso de la implementación. Tanto el paradigma Funcional como el Lógico tomaron un mayor tiempo en dominar, debido al cómo están pensados, sin el uso de variables ni estructurado.

En conclusión, se consiguió un mayor dominio del paradigma y se logró implementar una solución adecuada utilizando el paradigma orientado a objetos. El paradigma logra niveles de abstracción que pueden ser más complejos de implementar con otros paradigmas.

10. Referencias

Stroustrup, B. (1988) "What is object-oriented programming?," in *IEEE Software*, vol. 5, [Artículo] Extraído desde <https://ieeexplore.ieee.org/document/2020>

Gacitúa, D. (2023). *Paradigmas de programación. Paradigma Orientado a Objetos*. [Diapositivas de PowerPoint]. Extraído desde Moodle Usach Paradigmas de Programación (13204 y 13310) 1-2023

11. Anexos

Diagrama 1: Diagrama de Análisis

11. Anexos

Diagrama 1: Diagrama de Análisis

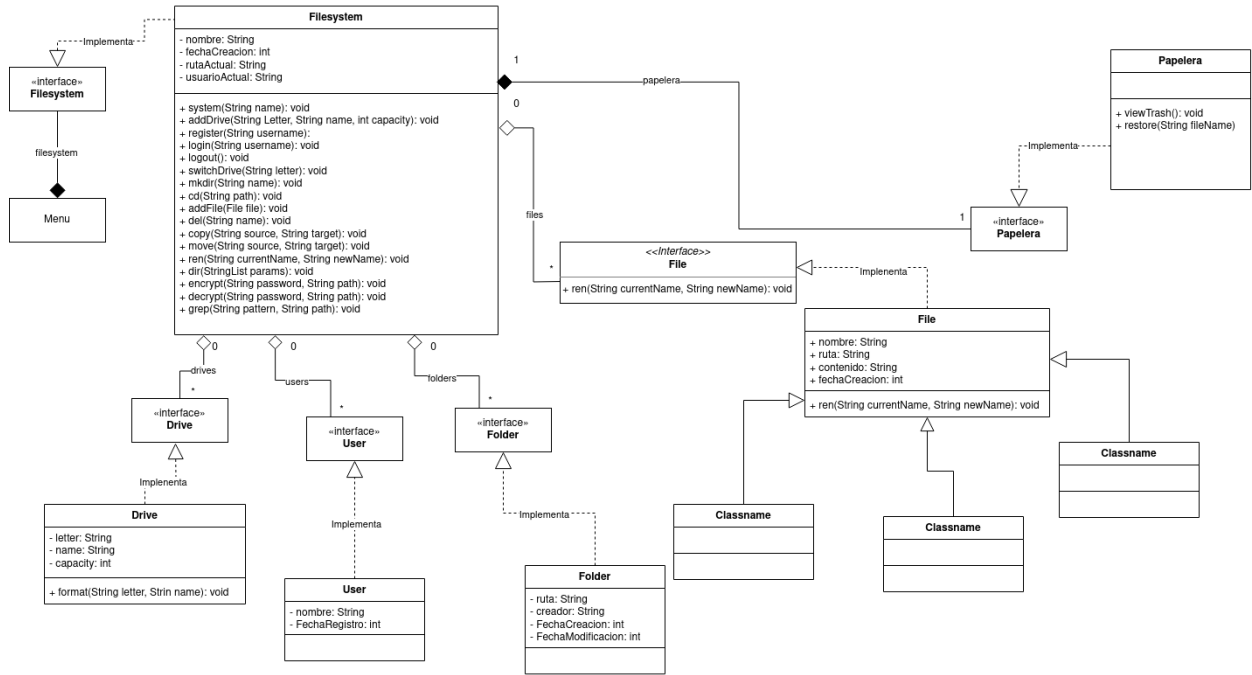


Diagrama 2: Diagrama de Diseño

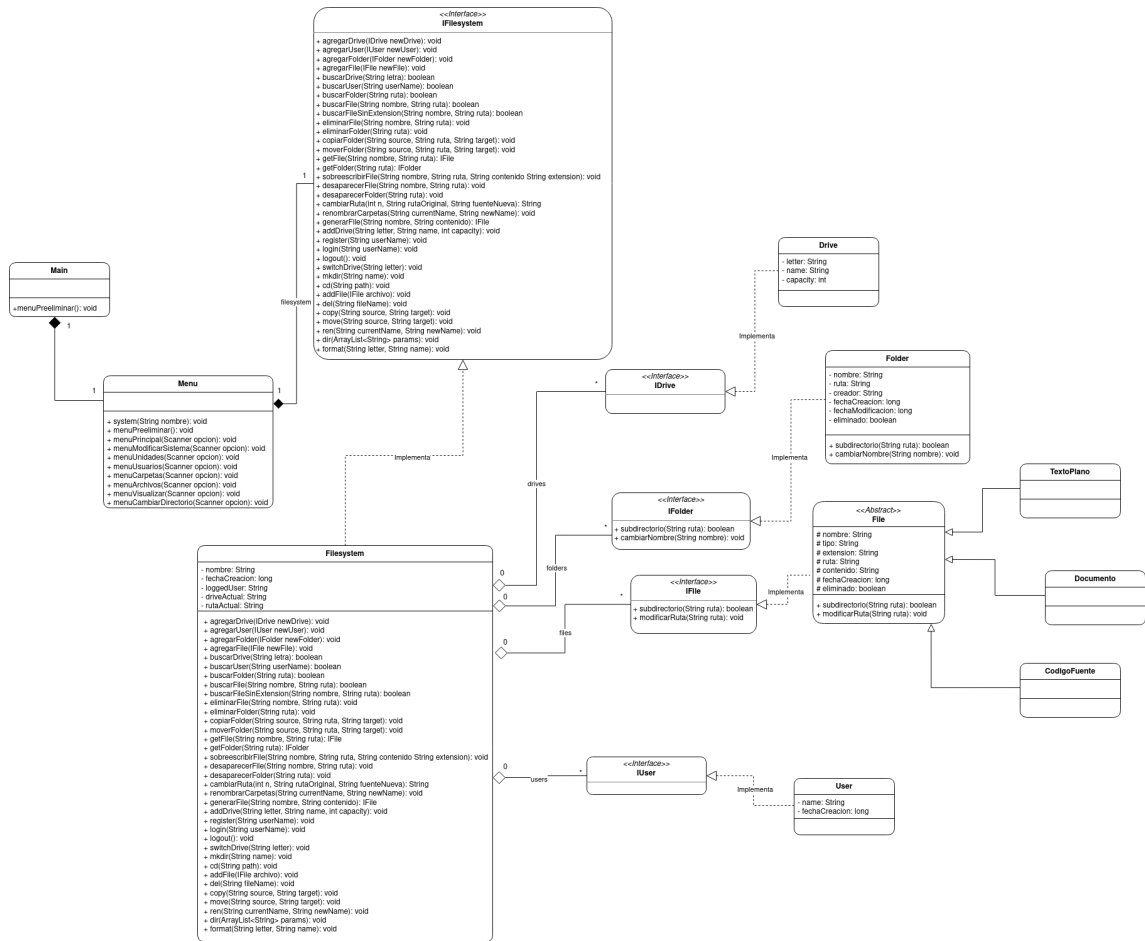
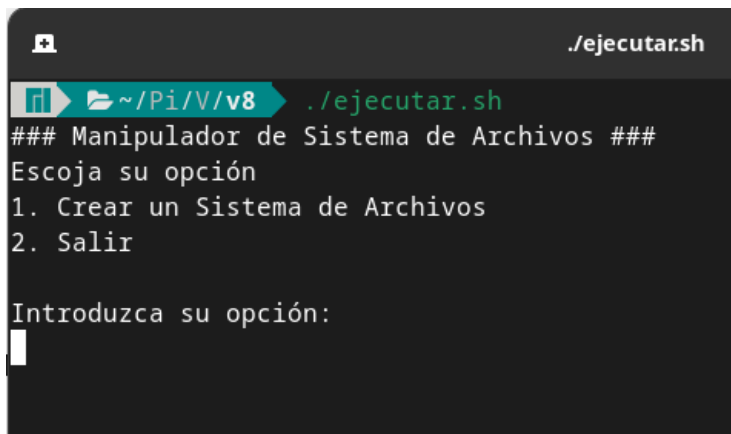


Figura 1: demostración “ejecutar.sh”.

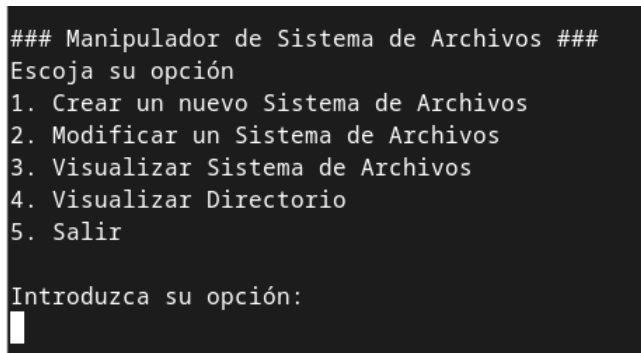


A terminal window with a dark background. The title bar at the top shows a window icon and the text `./ejecutar.sh`. The terminal content shows a prompt `~/Pi/V/v8` followed by the command `./ejecutar.sh` in green. The output is a menu in Spanish for a file system manipulator. It asks the user to choose an option from a list of two: '1. Crear un Sistema de Archivos' and '2. Salir'. Below the list, it says 'Introduzca su opción:' followed by a white cursor line.

```
## Manipulador de Sistema de Archivos ##
Escoja su opción
1. Crear un Sistema de Archivos
2. Salir

Introduzca su opción:
█
```

Figura 2: demostración del menú principal.



A terminal window with a dark background. The content shows a menu in Spanish for a file system manipulator. It asks the user to choose an option from a list of five: '1. Crear un nuevo Sistema de Archivos', '2. Modificar un Sistema de Archivos', '3. Visualizar Sistema de Archivos', '4. Visualizar Directorio', and '5. Salir'. Below the list, it says 'Introduzca su opción:' followed by a white cursor line.

```
## Manipulador de Sistema de Archivos ##
Escoja su opción
1. Crear un nuevo Sistema de Archivos
2. Modificar un Sistema de Archivos
3. Visualizar Sistema de Archivos
4. Visualizar Directorio
5. Salir

Introduzca su opción:
█
```

Figura 3: demostración intento de agregar carpeta.

```
### Manipulador de Sistema de Archivos ###
Escoja su opción
1. Crear un nuevo Sistema de Archivos
2. Modificar un Sistema de Archivos
3. Visualizar Sistema de Archivos
4. Visualizar Directorio
5. Salir

Introduzca su opción:
2

### Modificar un Sistema de Archivos ###
Escoja su opción
1. Unidades
2. Usuarios
3. Carpetas
4. Archivos
5. Cambiar Directorio
6. Salir

Introduzca su opción:
3

### Carpetas ###
Escoja su opción
1. Agregar Carpeta
2. Eliminar Carpeta
3. Renombrar Carpeta
4. Copiar Carpeta
5. Mover Carpeta
6. Salir

Introduzca su opción:
1
Debe fijar una unidad para crear un directorio.
```


Figura 4: demostración crear archivo con “addFile”.

```
### Modificar un Sistema de Archivos ###
Escoja su opción
1. Unidades
2. Usuarios
3. Carpetas
4. Archivos
5. Cambiar Directorio
6. Salir

Introduzca su opción:
4

### Archivos ###
Escoja su opción
1. Agregar Archivo
2. Eliminar Archivo
3. Renombrar Archivo
4. Copiar Archivo
5. Mover Archivo
6. Salir

Introduzca su opción:
1

Ingrese el nombre del archivo que desea crear:
fool.txt
Ingrese el contenido del archivo:
prueba
El archivo se agregó exitosamente.
```

Figura 5: demostración Visualizar Sistema de Archivos

```
### Manipulador de Sistema de Archivos ###
Escoja su opción
1. Crear un nuevo Sistema de Archivos
2. Modificar un Sistema de Archivos
3. Visualizar Sistema de Archivos
4. Visualizar Directorio
5. Salir

Introduzca su opción:
3

### Sistema ###

Filesystem: Gentoo
Fecha de creación: Mon Jul 17 22:14:26 CLT 2023
Usuario actual: user1
Ruta Actual: c:/

## Unidades ##

Unidad: c
Nombre: MEDIA
Capacidad: 222

Contenidos:
Carpetas:
  c:/      Mon Jul 17 22:15:51 CLT 2023      Mon Jul 17 22:16:22 CLT 2023
  Archivos:
    fool.txt  Texto Plano  prueba      Mon Jul 17 22:16:22 CLT 2023
#####

## Usuarios ##
user1      Mon Jul 17 22:15:57 CLT 2023

### Papelera ###
Carpetas:
Archivos:
```

Figura 6: demostración del método “dir”.

```
### Manipulador de Sistema de Archivos ###
Escoja su opción
1. Crear un nuevo Sistema de Archivos
2. Modificar un Sistema de Archivos
3. Visualizar Sistema de Archivos
4. Visualizar Directorio
5. Salir

Introduzca su opción:
4
Introduzca los parámetros:
/o N
c:/
ayudas
fool.txt
repositorio
tabla.pdf
```

Tabla 1: Alcances de los requerimientos funcionales.

Función	Alcance Conseguido
Clases y Estructuras	Alcance Completo
Menú Interactivo	Alcance Completo
system	Alcance Completo
addDrive	Alcance Completo
register	Alcance Completo
login	Alcance Completo
logout	Alcance Completo
switchDrive	Alcance Completo
mkdir	Alcance Completo
cd	Se implementó una versión simple del método
addFile	Alcance Completo
del	Se implementó una versión simple del método
copy	Alcance incompleto, algunos casos del comodín no funcionan el 100% de las veces.
move	Alcance Completo
ren	Alcance Completo
dir	Alcance Completo exceptuando "/o [-]D", "/a" y "/s".
format	Alcance Completo