Introducción

Este proyecto a grandes rasgos consiste en crear un filesystem en el lenguaje rust o C, este debe residir en el espacio de usuario y tiene como objetivo usar imágenes blanco y negro para almacenar archivos. Su información se almacena en los pixeles de color blanco y negro.
Es importante que este FileSystem tenga las siguientes funciones de la biblioteca FUSE: getattr, create, open, read, write, rename, mkdir, readdir, opendir, rmdir, statfs, fsync, access, unlink, flush y iseek.
Como máximo debe usar 1000x1000 px. El sistema debe usar i-nodos como estructura de indexación.

Debe contener los siguientes archivos:
● Mkfs.bwfs (Crea el FS)
● Fsck.bwfs (Realiza un chequeo de consistencia del BWFS)
● Mount.bwfs (Lo monta en el Sistema Operativo)
Es importante destacar que el FS debe ser persistente en el disco duro del equipo, también debe ser posible crear archivos de cualquier tipo y de cualquier tamaño. (Mientras no se pase de 1000 x 1000 px)
Por último se debe implementar una estrategia para administrar la fragmentación del sistema de archivos.

Ambiente de desarrollo

- **Ubuntu 22.04 LTS**
- **CLion 2022.2**
- **GitHub**
- **Rust versión 1.63.0**

Estructuras de datos usadas y funciones

mkfs.bwfs (FileSystem Make)

Para la creación del FileSystem se usó:

fileStructure:

- FileAttrDef (Struct): Este struct define los parámetros o atributos que va a tener cada archivo o directorio.
- TimeSpecDef (Struct): Define los atributos de tiempo que va a tener un archivo o directorio.
- FileTypeDef (Enum): Define los tipos que va a tener un archivo o directorio.

fsstructure:

- **Disk**
    - Disk (Struct): Estructura que define los atributos del disco en el que se va a guardar los datos del FS.
    - new (Implementación de Disk): Crea un nuevo Disk, crea un nuevo superbloque de memory blocks, asigna los tiempos iniciales y le da atributos.

    - get_next_available_inode (Implementación de Disk): Obtiene el siguiente inode que este disponible en el Disk.
    - write_inode (Implementación de Disk): Escribe el inode ingresado al superbloque de inodes.
    - remove_inode (Implementación de Disk): Elimina el inode ingresado del superbloque de inodes.
    - clear_reference (Implementación de Disk): Elimina la referencia a un bloque de memoria de un inode.
    - add_reference (Implementación de Disk): Agrega la referencia a un bloque de memoria de un inode.
    - get_inode (Implementación de Disk): Obtiene un inode por medio del Id, si no encuentra nada no devuelve nada.
    - get_mut_inode (Implementación de Disk): Obtiene un inode mutable por medio del Id, si no encuentra nada no devuelve nada.
    - find_inode_by_name (Implementación de Disk): Obtiene un Inode por medio del nombre, si no encuentra nada no devuelve nada.
    - fill_memory_block (Implementación de Disk): Agrega datos a un bloque de memoria asociado a un Inode buscado por medio del Id.
    - delete_data_to_memory_block (Implementación de Disk): Borra datos a un bloque de memoria asociado a un Inode buscado por medio del Id.
    - write_content (Implementación de Disk): Escribe los datos en el memory block asociado a un Inode por medio del Id.
    - get_bytes_content (Implementación de Disk): Obtiene los datos en el memory block asociado a un inode por medio del Id.
- **filesystem_management**
    - BWFS (Struct): Define la estructura del FS, básicamente que disco va a usar.
    - new (Implementación de BWFS): Crea un nuevo FS basado en un disco en especifico.
    - load (Implementación de BWFS): Carga el nuevo disco al FS.
    - get_disk (Implementación de BWFS): Obtiene el disco que se esta usando en el FS actualmente.
    - set_disk (Implementación de BWFS): Setea el nuevo disco sobre el que se va a basar el FS.
    - save_fs (Implementación de BWFS): Guarda el FileSystem en una imagen blanco y negro.
    - drop (Implementación de Drop para BWFS): Apaga el FS guardandolo de manera persistente en imagenes.
    - getattr (Implementación de Filesystem para BWFS): Obtiene los atributos de un archivo existente en el FS.
    - create (Implementación de Filesystem para BWFS): Crea un archivo nuevo en el FS.
    - open (Implementación de Filesystem para BWFS): Abre un archivo existente en el FS.
    - read (Implementación de Filesystem para BWFS): Lee un archivo existente en el FS.
    - write (Implementación de Filesystem para BWFS): Escribe sobre un archivo ya existente en el FS.
    - rename (Implementación de Filesystem para BWFS): Renombra el archivo existente en el FS.
    - mkdir (Implementación de Filesystem para BWFS): Crea un directorio nuevo en el FS en el que se pueden guardar archivos.
    - readdir (Implementación de Filesystem para BWFS): Lee el directorio que se le pase como parámetro, este debe existir en el FS.

    - opendir (Implementación de Filesystem para BWFS): Abre un directorio existente en el FS.
    - rmdir (Implementación de Filesystem para BWFS): Elimina o remueve un directorio existente en el FS.
    - statfs (Implementación de Filesystem para BWFS): Muestra las estadísticas básicas del FS, como cantidad de inodos o bloques de memoria.
    - fsync (Implementación de Filesystem para BWFS): Sincroniza los contenidos de los archivos, si es diferente a 0 no borra los metadatos pero si los datos del usuario.
    - access (Implementación de Filesystem para BWFS): Revisa si puede acceder a un archivo ya existente en el FS.
    - unlink (Implementación de Filesystem para BWFS): Desvincula un archivo, ya sea vinculo normal o vínculo simbólico.
    - flush (Implementación de Filesystem para BWFS): Trata de eliminar o *flushear* los datos del caché.
    - lseek (Implementación de Filesystem para BWFS): Encuentra el primer hueco de datos en un offset especifico.
- **I-node**
    - I-node (Struct): Define los elementos que debe contener un I-node como lo son el nombre, sus atributos y referencias a bloques de memoria.
    - add_reference (Implementación de I-node): Agrega una referencia a un bloque de memoria a un I-node.
    - delete_reference (Implementación de I-node): Elimina una referencia a un bloque de memoria a un I-node.
    - change_name (Implementación de I-node): Cambia el nombre de un I-node.
- **memory_block**
    - MemoryBlock (Struct): Define la estructura de un bloque de memoria, que tiene una referencia a su I-node padre y los datos.
    - add_data (Implementación de MemoryBlock): Agrega datos a un bloque de memoria, aquí se realiza la verificación de que este no pase de 1000*1000 de tamaño.
    - delete_data (Implementación de MemoryBlock): Elimina todos los datos de un bloque de memoria.
- **save_disk**
    - encode (Función): Codifica un disco y lo serializa a binario.
    - decode (Función): De-codifica un disco y lo de-serializa de binario.
    - write_pixels (Función): Se encarga de escribir los datos serializados en binario de un disco a una imagen de blanco y negro.
    - validate_path (Función): Valida que la ruta de la imagen a escribir exista.
    - validate_fs_path (Función): Valida que la ruta del fs en imagenes exista, que sea cargable en el FileSystem.
    - load_disk (Función): Carga un disco usando la ruta de este.

fsck.bwfs (FileSystem Check)

En este caso se utilizaron las siguientes funciones:

- **save_disk**
    - validate_path (Función): Valida que la ruta de la imagen a escribir exista.
    - load_disk (Función): Carga un disco usando la ruta de este.

mount.bwfs (FileSystem Mount)

En el mount se utilizaron las funciones:

- **filesystem_management**
  - new (Implementación de BWFS): Crea un nuevo FS basado en un disco en especifico.
  - load (Implementación de BWFS): Carga el nuevo disco al FS.
- **save_disk**
  - validate_fs_path (Función): Valida que la ruta del fs en imagenes exista, que sea cargable en el FileSystem.
  - load_disk (Función): Carga un disco usando la ruta de este.

Instrucciones para ejecutar el programa

1. Se debe asegurar de tener los 3 binarios:
   - fsck.bwfs
   - mount.bwfs
   - mkfs.bwfs
2. En caso de querer **revisar el estado de un FS** existente se debe:
   2.1 Usar el binario fsck.bwfs de la siguiente manera por medio de la terminal: `./fsck.bwfs folderdelfs/`
3. En caso de querer **montar un FS** existente debe:
   3.1 Revisar que el FileSystem este en buenas condiciones, esto se puede realizar usando el binario fsck.bwfs de la siguiente manera por medio de la terminal: `./fsck.bwfs folderdelfs/`
   3.2 Después de confirmar el correcto estado del FS debe montarlo usando el binario mount.bwfs de la siguiente manera por medio de la terminal: `./mount.bwfs folderdelfs/ mountpoint/` con el parámetro mountpoint siendo una dirección en la que se quiera montar.
4. Por último en caso de querer **crear un nuevo FS** se debe:
   4.1 Crear un FS desde cero usando el binario mkfs.bwfs de la siguiente manera por medio de la terminal: `./mkfs.bwfs folderdelnuevofs/`
   4.2 Después se debe revisar que el FileSystem este en buenas condiciones, esto se puede realizar usando el binario fsck.bwfs de la siguiente manera por medio de la terminal: `./fsck.bwfs folderdelfs/`
   4.3 Después de confirmar el correcto estado del FS debe montarlo usando el binario mount.bwfs de la siguiente manera por medio de la terminal: `./mount.bwfs folderdelfs/ mountpoint/` con el parámetro mountpoint siendo una dirección en la que se quiera montar.

Actividades realizadas por estudiantes

Fecha	Inicio	Fin	Avance Realizado	Estudiante
5/11/2022	12:00	16:00	Se trabajo en la estructura del proyecto	Todos
7/11/2022	14:00	20:28	Compila la estructura inicial	Todos
8/11/2022	12:00	15:00	Compila la estructura inicial con agregados	Jose Pablo

Fecha	Inicio	Fin	Avance Realizado	Estudiante
8/11/2022	17:00	20:20	Compila el FileSystem, primeras nociones	Jose Pablo y Luis
12/11/2022	16:00	21:42	Se empieza el sistema Black and White con las imagenes	Todos
14/11/2022	10:00	13:27	Imagen Funcional	Luis
16/11/2022	17:00	21:27	Cambios generales y que no pase de 1000x1000	Pablo
17/11/2022	18:00	21:49	Funciones generales y guardado de imagenes	Todos
18/11/2022-19/11/2022	8:00 (Dia 18)	11:35 (Dia 19)	Guarda el FS en imagenes	Todos
19/11/2022	17:00	18:34	Creacion y division de los modulos	Pablo
21/11/2022	10:00	11:49	Implementacion de algunos metodos	Pablo y Luis
21/11/2022	11:00	15:15	Metodos adicionales, falta prueba	Todos
22/11/2022	15:00	21:38	Creacion de la documentacion y documentacion interna	Pablo
22/11/2022	21:38	21:41	Update de documentacion	Pablo
22/11/2022	23:00	23:55	Documentacion al dia y lecciones aprendidas	Pablo
23/11/2022	12:00	15:46	Persistencia	Todos
23/11/2022	19:00	22:25	Detalles Finales al FS	Todos
23/11/2022	22:25	22:55	Modulos Funcionales	Pablo
24/11/2022	11:00	13:00	Actualizacion de documentacion	Luis y Pablo
24/11/2022	13:00	16:35	Refactorizacion y pulido	Luis
24/11/2022	17:00	21:00	Entrega Final	Todos

Nota: Para el trabajo en conjunto se utilizo la herramienta Code With Me que viene integrada en el IDE CLion 2022.2.

GitLog

```
commit c2ddfd33df6e6e5bafed1c6876ebf191ccdaee59 (HEAD -> main, origin/main,
origin/HEAD)
Author: JP171001 <josepa1710@gmail.com>
```

```
Date:   Thu Nov 24 21:05:59 2022 -0600

    entrega final

commit 583b0f7de04f70551b08a9d9c24c53155a319fed
Author: JP171001 <josepa1710@gmail.com>
Date:   Thu Nov 24 18:07:45 2022 -0600

    casi terminado x3

commit 0aeba841e67eff16a0d8392339e4ec4691a464ac
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Thu Nov 24 16:32:04 2022 -0600

    refactored and cleaned mtfs.bwfs

commit 96b8ee4f7e3639b030d7680df3fd73c8707d71c6
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Thu Nov 24 15:54:49 2022 -0600

    refactored and cleaned fsck.bwfs

commit 963b6fd5e1f5b821452c9f2d8c6b94b251e02156
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Thu Nov 24 15:12:20 2022 -0600

    refactored and cleaned mkfs.bwfs

commit 2b15f92efff06c6b47831ee44c9775fcf4d0d5fe
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Thu Nov 24 12:41:15 2022 -0600

    Sin proyecto 2 y docu actualizada

commit 114df117adf48d8caf3b3bfbaeab69131ed6de20
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Wed Nov 23 22:58:27 2022 -0600

    Without Targets

commit 7aaa0319fba56b43ff7cdb8a7ea44afab5d77379
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Wed Nov 23 22:57:28 2022 -0600

    Update .gitignore

commit 106270b2645bc8dc6031f2421b45bd31c8f83945
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Wed Nov 23 22:55:36 2022 -0600

    Modulos completos

commit ac42c8c5dee2d31736643fca579b56e7732a7f9c
Author: JP171001 <josepa1710@gmail.com>
Date:   Wed Nov 23 22:25:23 2022 -0600
```

```
    casi terminado x2

commit bb999ecd1bb5d0537ffd015a7aafd545ffd25156
Author: JP171001 <josepa1710@gmail.com>
Date:   Wed Nov 23 22:05:40 2022 -0600

    casi terminado

commit c10f0435878a7ffca62e59ff0d8aecfca028e702
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Wed Nov 23 15:46:19 2022 -0600

    Persistencia y demás

commit 0e32a70d1f1ca44851df7c4aefbd78086eaf7b7a
Merge: a6eb646 bd1ed00
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Wed Nov 23 00:00:25 2022 -0600

    Merge branch 'main' of https://github.com/pabloamh27/FileSystem into main

commit bd1ed006f0f4d00be0de537088f34a7a0a697087
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Tue Nov 22 23:56:42 2022 -0600

    Lecciones aprendidas

commit 4eb9d1c5592faaecb583f46c5078ffe611bd405e
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Tue Nov 22 23:55:42 2022 -0600

    Documentación Completa al día

commit 7332b909ef8ba252f6f84bc97e8ffc538cfd7f94
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Tue Nov 22 21:41:34 2022 -0600

    Update de docu interna

commit f88708e4b3e021d54efbbe5eeed2e58ab0ef5bab
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Tue Nov 22 21:38:12 2022 -0600

    Documentacion interna y externa

commit e1aba2cb85edb541a822cc4bfb811311fe448181
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Tue Nov 22 17:44:11 2022 -0600

    lookup problems

commit 454feb38a61e7248bf201889666bbdd6421b5aa1
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Mon Nov 21 15:15:21 2022 -0600
```

```
    Algunos metodos más, pero falta probarlos bien

commit 9537cccbcf4929d6207cd61824b122dfecb34ef6
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Mon Nov 21 11:52:24 2022 -0600

    Implementación de algunos de los métodos

commit a6eb64663a1510a6d76ccb739d2acadbd5544280
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Sat Nov 19 18:34:18 2022 -0600

    Division para hacer los binarios

commit 1883fb7abba0827a5c64cb80367bf8a1eb35a8ba
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Sat Nov 19 11:35:57 2022 -0600

    se corrige el commit anterior

commit e7bc87112e26d6d3945664100b87abd985d14fd9
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Sat Nov 19 11:34:13 2022 -0600

    se corrige el commit anterior

commit 049a1655353f1100b91cafceba773dc57a8ed352
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Fri Nov 18 10:17:47 2022 -0600

    Ya guarda el fileSystem en imagenes

commit 990a1fda220ad950a161373fdb1e27f76a508661
Author: JP171001 <josepa1710@gmail.com>
Date:   Thu Nov 17 21:49:14 2022 -0600

    guarda imagenes y demás

commit 93fbef87901aaba4805780d6701acaaf3d2484ce
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Wed Nov 16 21:39:27 2022 -0600

    Verifica que no pase 1000x1000

commit ccef0f9d0201c923332841863c57f0953c455af3
Author: LuisAndresTEC <lrojasmurillo7@estudiantec.cr>
Date:   Mon Nov 14 13:27:12 2022 -0600

    Ya funciona la imagen

commit aa29fa2966792ca9e0efee42acacf84b0e588a07
Author: JP171001 <josepa1710@gmail.com>
Date:   Sat Nov 12 21:42:48 2022 -0600
```

```
    comienza el black and white

commit 23b271e2f3db1d1ab85592241f29d1835bddab6f
Author: JP171001 <josepa1710@gmail.com>
Date:   Tue Nov 8 20:19:51 2022 -0600

    compilando estructura inicial 3

commit 975c813b5b2c44bb0e08ae4a1e29c76a28c3fe4e
Author: Jose Pablo Quiros Hidalgo <70715261+JP171001@users.noreply.github.com>
Date:   Tue Nov 8 16:06:38 2022 -0600

    Create .gitignore

commit f3f6fa23e323e50dfb8b20439f47b4fce4e68bda
Author: JP171001 <josepa1710@gmail.com>
Date:   Tue Nov 8 14:59:25 2022 -0600

    compilando estructura inicial 2

commit 613cefd5c24f5e3cc41474f782cee634a2a09cd1
Author: JP171001 <josepa1710@gmail.com>
Date:   Mon Nov 7 20:28:06 2022 -0600

    compilando estructura inicial

commit 060895d502f62cead5626cd5fd73c373e3c5eefd
Author: JP171001 <josepa1710@gmail.com>
Date:   Sat Nov 5 16:30:38 2022 -0600

    proyecto iniciado

commit 1e91c512769b67fe08e7f065c8c37e1c4eb13696
Author: Pablo Munoz Hidalgo <53487847+pabloamh27@users.noreply.github.com>
Date:   Tue Nov 1 13:47:35 2022 -0600

    Initial commit
```

Autoevaluación

Evaluación	Nota máxima obtenible	Nota Autoevaluada
mkfs.bwfs	14%	14%
fsck.bwfs	5%	5%
mount.bwfs	10%	10%
Funciones de la biblioteca	26%	25% con los extras
Documentación	20%	20%
Persistencia en el Disco	25%	25%

Autoevaluación	José Pablo	Pablo Muñoz	Luis Andrés
Aprendizaje de mkfs	5	5	5
Aprendizaje de fsck	5	5	5
Aprendizaje de mount	5	5	5
Aprendizaje de implementación de funciones	5	5	5
Aprendizaje de Diseño de FileSystem	5	5	5

Lecciones Aprendidas

En este proyecto se trató mucho el tema de como un SO administra los datos por medio de un FileSystem, se aprendió que es fundamental tener un buen FS para poder administrar los datos de manera eficiente, rápida y persistente, sin embargo no que sea fundamental no quiere decir que sea una tarea fácil, como grupo la pasamos especialmente difícil con este proyecto dado a que además de estar a final de semestre este lo obliga como ingeniero a pensar fuera de la caja.

Es un proyecto que enseña de manera espectacular como funciona el sistema de los i-nodos que (en este caso) fue el que escogimos para el proyecto, también es curioso ya que le enseña al programador y le brinda un mejor conocimiento sobre lo que está pasando por debajo de la interfaz gráfica que vemos todos los días.

En resumen, una conclusión muy buena a todos los temas vistos a lo largo del curso ya que da una mejor perspectiva sobre el sistema operativo y como este esta estructurado a lo interno, además agradecemos el uso de Rust ya que demuestra que es un lenguaje perfecto para la programación a bajo nivel, es un lenguaje super completo y definitivamente es de las cosas que más rescatamos del curso.

Bibliografía

- “image - Rust,” docs.rs. https://docs.rs/image/latest/image/ (accessed Nov. 24, 2022).
- H. Uddin, “Decoding and encoding images in Rust using the image crate,” LogRocket Blog, May 24, 2022. https://blog.logrocket.com/decoding-encoding-images-rust-using-image-crate/?ref=morioh.com&utm_source=morioh.com (accessed Nov. 24, 2022).
- “Social Network for Programmers and Developers,” morioh.com. https://morioh.com/p/a3e5136ef8db (accessed Nov. 24, 2022).
- Berner, Christopher. “FUSE (Filesystem in Userspace) for Rust.” GitHub, 21 Nov. 2022, github.com/cberner/fuser. Accessed 24 Nov. 2022.
- “File System - Rust Cookbook.” Rust-Lang-Nursery.github.io, rust-lang-nursery.github.io/rust-cookbook/file.html. Accessed 24 Nov. 2022.
- “Filesystem Operations - Rust by Example.” Doc.rust-Lang.org, doc.rust-lang.org/rust-by-example/std_misc/fs.html. Accessed 24 Nov. 2022.
- “Libfuse: Fuse_operations Struct Reference.” Libfuse.github.io, libfuse.github.io/doxygen/structfuse__operations.html. Accessed 24 Nov. 2022.
- Neuhaus, Andreas. “Rust FUSE - Filesystem in Userspace.” GitHub, 18 Nov. 2022, github.com/zargony/fuse-rs. Accessed 24 Nov. 2022.
- “Png - Rust.” Docs.rs, docs.rs/png/latest/png/. Accessed 24 Nov. 2022.

- “Std::Fs - Rust.” Doc.rust-Lang.org, doc.rust-lang.org/std/fs/. Accessed 24 Nov. 2022.