Instituto Tecnologico de Costa Rica

Sistemas Operativos

Proyecto II

Black & White File system

Jean Marco Rojas

2015040717

Jonathan Guillén

201261579

Profesor
Sr. K. Moraga

Junio 26, 2021

1 Introduction

BWFS es un sistema de archivos que reside en el espacio de usuario. Este sistema de archivos tiene como objetivo utilizar los imágenes blanco y negro para almacenar archivos. Para BWFS el espacio físico donde se almacena su información y toda su estructura se encuentra en pixels de color blanco y negro, definidos en la creación del FS. Se realizará la re-implementación de las siguientes funciones utilizando la biblioteca FUSE:

- getattr
- create
- open
- read
- write
- rename
- mkdir
- readdir
- opendir
- rmdir
- statfs
- fsync
- access
- unlink
- flush
- lseek

2 Ambiente de desarrollo:

Cada miembro del equipo trabajara desde su propia máquina para la realización del código necesario para el proyecto, el problema presentado se dividirá en asignaciones y cada miembro seleccionará de cuales se hará cargo distribuyendo equitativamente las cargas de trabajo, esto se hará mediante sprints semanales, que nos permite dividir y observar el trabajo realizado por los integrantes del grupo, cada vez que se dé por terminada una asignación o parte de la misma se debe subir el trabajo realizado a Github de manera tal que permita el acceso a dicha información a los demás miembros del grupo, esto con el fin de tener

una integración continua con versiones del trabajo realizado, poder realizar revisiones y comentar sobre el código realizado, además al final de cada sprint se hace una autoevaluación para detectar las debilidades del grupo de trabajo y así saber en qué mejorar. A continuación se detallan las herramientas utilizadas para la realización de esta tarea:

C: En el caso de este lenguaje se utilizará el compilador gcc y se programara utilizando Sublime text 3, Clion, VScode y Netbeans 8.2.

Latex: Este editor de texto será la herramienta principal para la realización de los documentos del kick off y la documentación final de la tarea.

GitHub: Como sistema de control de versiones.

Google Chrome: Como navegador web y donde se correrá el servidor, en su versión 89.0.4389.114.

Ubuntu: Como sistema operativo donde será desarrollada la tarea, en su versión 20.04.2.0.

Fuse: Biblioteca usada en c para la creación del file system

3 Estructuras de datos usadas y funciones

struct: inode:: Esta estructura representa un fichero en el FS. Cada i node contiene todos los atributos que caracterizan a un fichero (permisos, owner, grupo, nombre, path, etc).

struct: superBloque: Esta estructura contempla toda la información de nuestro FS y un puntero al punto de montaje del FS.

loadBMP:: Esta función se encarga de leer la imagen BMP y pasar la información al disco (imgdata, en superBloque).

saveBMP: Esta función baja la información del disco para guardar la imagen BMP con el trabajo de esteganografía hecha.

file_to_sb:: Para evitar errores y facilitar el acceso y la lectura, se replica constantemente la lista de índices (i nodes) que se encuentran en super_bloque a la lista global lista inodes.

actualizar: Se encarga de recordar la estructura del chero (el árbol de i nodes) en un le, Estructura.dat.

4 Instrucciones para ejecutar el programa

Para correr el programa con correctitud es necesario contar con una imagen BMP 1920x1080. Además de haber creado un file que sirva como punto de montaje para ImageFS. Para compilar el proyecto es necesario ejecutar el siguiente comando: "gcc -Wall imageFS.c -lm ‘pkg-config fuse –cflags –libs‘ -o fs"

Para ejecutar debemos asignar nuestro punto de montaje: ./fs -f ./[punto de montaje]

5 Actividades realizadas

- 1 de junio 2021 (Jean Marco, Jonathan): Buscar información sobre el funcionamiento de Fuse (3 horas).
- 1 de junio 2021 (Jean Marco): Investigación sobre esteganografía (2 horas).
- 3 de junio 2021 (Jean Marco): Investigación sobre la creación de le system en C (4 horas).
- 4 de junio 2021 (Jean Marco): Investigación sobre FUSE e i-nodos (5 horas).
- 8 de junio 2021 (Jean Marco, Jonathan): Investigación de formato RAW y BMP (1 hora).
- 9 de junio 2021 (Jonathan): Investigación sobre las crear imagenes en c (2 horas).
- 10 de junio 2021 (Jonathan): Investigación sobre creae las imagenes formato bmp (2 horas).
- 12 de junio 2021 (Jonathan): Investigación sobre guardar el bmp (3 horas).
- 15 de junio 2021 (Jean Marco): Programación errónea de las funciones Open, (3 horas).
- 16 de junio 2021 (Jean Marco): Programación errónea de la función para guardar el BMP (4 horas).
- 18 de junio 2021 (Jonathan): Investigación sobre cargar el bmp (3 horas).
- 19 de junio 2021 (Jean Marco): Investigar sobre BMP en c (3 horas).
- 20 de junio 2021 (Jonathan): Programación errónea de la función para cargar el BMP (4 horas).
- 22 de junio 2021 (Jonathan): Programación errónea de la función para guardar y cargar el BMP utilizando un api (4 horas).
- 24 de junio 2021 (Jonathan): Programación errónea de la función para guardar y cargar el BMP (4 horas).
- 25 de junio 2021 (Jean Marco): Programación errónea de la función para cargar el BMP (4 horas).
- 25 de junio 2021 (Jonathan): Programación de la función para cargar el BMP (4 horas).

- 26 de junio 2021 (Jonathan): Programación de la errónea función para guardar el BMP (4 horas).
- 26 de junio 2021 (Jean Marco, Jonathan): Desarrollo de la documentación.

6 Autoevaluación

El proyecto quedó incompleto, falto el desarrollo de mkfs, fsck y una parte de mount. Como el mayor de los problemas, podemos reconocer la falta de información disponible para realizar investigación. Además, la curva de aprendizaje fue bastante mayor de lo esperado. Como principales limitaciones encontramos que: la imagen debe ser de 1920x1080, se contempla hasta un máximo de 100 files, el tamaño del bloque es de un byte, cada cuatro píxeles hay un solo byte de información útil. Al trabajar con una imagen de 1920x1080 pixeles, si lo dividimos entre los píxeles que necesitamos para obtener un byte de información, podemos encontrar que contamos con 518 400 bytes totales que nos funcionan como disco de almacenamiento.

6.1 Evaluación

- File system: Completado
- Documentación: Completado
- Kick-off: Completado

6.2 Autoevaluación

Jean Marco:
[5] Aprendizaje de mkfs.
[5] Aprendizaje de fsck.
[2] Aprendizaje de mount.
[5] Aprendizaje de implementacion de funciones.
[5] Aprendizaje de Diseño de Filesystem.

Jonathan:
[5] Aprendizaje de mkfs.
[5] Aprendizaje de fsck.
[2] Aprendizaje de mount.
[3] Aprendizaje de implementacion de funciones.
[5] Aprendizaje de Diseño de Filesystem.

7 Lecciones Aprendidas y Recomendaciones

- Investigar y leer la documentación de FUSE, a su vez de ver tutoriales en youtube.
- Funcionamiento de un file system a profundidad.
- Investigar sobre cómo crear su propia biblioteca.
- Organizarse desde el inicio, por la magnitud del proyecto.
- Investigar el punto de montado e imagenes bmp.

8 Bibliografía

- Writing a Simple Filesystem Using FUSE in C. (2016). Retrieved 27 June 2021, from https://www.maastaar.net/fuse/linux/filesystem/c/2016/05/21/writing-a-simple-filesystem-using-fuse/
- CS135 FUSE Documentation . (2021). Retrieved 27 June 2021, from https://www.cs.hmc.edu/ geoff/classes/hmc.cs135.201109/homework/fuse/fuse_doc.html
- An introduction to Linux filesystems. (2021). Retrieved 27 June 2021, from https://opensource.com/life/16/10/introduction-linux-filesystems