Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

Principios de sistemas operativos

QRFS

Desarrollado por

- #### Kevin Segura Rojas (2017153767)
- #### Jonathan Guillén (201261579)
- #### Jorge Gatgens Villalobos (2017110109)

Alajuela, II-S 2020

Introdución

En este proyecto se realizará la reimplementación de varias funciones del file system. Los archivos que se manejaran seran archivos QR, los cuales seran montados en el sistema operativo para su almacenamiento y manejo de los mismos. El proyecto esta compuesto por las siguientes partes: - QRFS: Son las funciones que se va a reimplementar y las cuales se encargan del manejo de los archivos. - mkfs.qrfs: Es el medio por el cual se crea el sistema de archivos y define con cual es el primer archivo de la particion. - fsck.qrfs: Realiza el checkeo del QRFS - mount.qrfs: Es el medio por el cual se montan los archivos en memoria para su almacenamiento

Ambiente de Desarrollo

- CLion: Mediante estos editores se realizó el código de todo el proyecto y el debbugeo necesario.
- GCC: mediante esta herramienta compilamos el código de proyecto.
- Ubuntu Linux: Sistema operativo en el que se desarrolló todo el proyecto.
- Github: Utilizamos este servicio como alojamiento de nuestro repositorio del proyecto.

Estructuras de datos usadas y funciones

- Inodes
- Pila de datos
- Listas

Librerías utilizadas

- PNG-dev
- libqrencode-dev
- String.h
- fuse.h

Funciones QRFS

Funciones a reimplementar (my_create, my_open, my_read, etc)

Funciones básicas de los file systemas solicitadas en el proyecto.

buscar_archivo()

Función que busca el inodo mediante el path.

find_i_father()

Función que busca el inodo padre del inodo actual mediante el path.

Qr_admin

leerQR()

Función que lee los datos de un qr.

QRdeBuffer()

Genera cuantos qr's sean necesarios de acuerdo al tamaño de la entrada

createQR()

Crea un qr con el nombre y datos específicados.

guardarPila()

Función que guarda la pila en disco usando qr.

restablecerPila()

Función que carga la pila desde unos qr en disco.

Instrucciones para ejecutar el programa

Para usar QRFS

gcc -Wall Qr_Admin.c pilaQR.c QRFS.c -lm -lpng -lqrencode pkg-config fuse --cflags --libs -o fs

./fs -f [punto de montaje]

Además, como paso extra hay que usar otra terminal para ingresar y utilizar el file system montado.

Actividades realizadas por estudiante

Jonathan Guillen

Actividad	Fecha	Tiempo
Estudiar sobre file systems.	18/01/2021	3 horas
Buscar información sobre inodes.	19/01/2021	2 horas
Creación prototipo de los inodes.	23/01/2021	4 horas
Creación de pila para el manejo de la fragmentación.	24/01/2021	6 horas
Escribir la Documentación.	31/01/2021	1 hora

Horas Totales: 16 horas.

Kevin Segura

Actividad	Fecha	Tiempo
Estudio sobre sistemas de archivos.	17/01/2021	3 horas
Estudio sobre la libreria fuse	18/01/2021	4:30 horas
Busqueda de como crear y leer codigos qr.	19/01/2021	5 horas
Se siguio con el estudio de los qr.	21/01/2021	4 horas
Más estudio de qr.	22/01/2021	4 horas
Reestruturación de algunos detalles de la pila.	25/01/2021	5 horas
Creación de las funciones necesarias para el manejo de qr.	26/01/2021	4 horas
Intento de implementación de Inodes con qr.	26/01/2021	5 horas

Actividad	Fecha	Tiempo
Se agrego funcionalidad de la pila para que funcionara de qr.	27/01/2021	3 horas
Más funciones necesarias para el uso de qr en el file system.	28/01/2021	4 horas
Más intentos de pegar el file system con las funciones de qr.	28/01/2021	5 horas
Creación de la documentación.	31/12/2021	1 hora

Horas Totales: 42:30 horas.

Jorge Gatgens

Actividad	Fecha	Tiempo
Estudio sobre sistemas de archivos.	15/01/2021	3 Horas
Estudio sobre los qr.	17/01/2021	3 horas
Investigación sobre las funciones que hay que reimplementar.	18/01/2021	5 Horas
Estudio de la librería fuse.	20/01/2021	2 Horas
Se continuo el estudio de la librería fuse.	21/01/2021	3 Horas
Implementación de las funciones en memoria.	22/01/2021	4 Horas
Se siguio con la implementación del file sistem en memoria.	24/01/2021	3 Horas
Preparando el file sistem para implementar el guardado por qr.	25/01/2021	4 Horas
Intentos de pegar el file system con las funciones de qr.	27/01/2021	5 Horas
Más intentos de pegar el file system con las funciones de qr.	28/01/2021	6 horas
Creación de la documentación	31/01/2021	1 horas

Horas Totales: 40 horas.

Evaluación

- mkfs.qrfs: 10 % de 14 %
- fsck.qrfs: 0 % de 5 %
- mount.qrfs: 0 % de 15 %
- Funciones de la biblioteca: 22 % de 26%.
- Documentación: 15 % de 15 %
- Implementación del FS: 14 % de 30 %.
- Diseño del FS: 10 % de 10 %
- Manejo de Fragmentación: 10 % de 15 %

Autoevaluación

Jonatha Guillen

[1] [2] [3] [3] [5] Aprendizaje de mkfs. [2]

[1] [2] [3] [3] [5] Aprendizaje de fsck. [3]

[1] [2] [3] [3] [5] Aprendizaje de mount. [2]

[1] [2] [3] [3] [5] Aprendizaje de implementacion de funciones. [3]

[1] [2] [3] [3] [5] Aprendizaje de Diseño de Filesystem. [3]

Kevin Segura

[1] [2] [3] [3] [5] Aprendizaje de mkfs. [3]

[1] [2] [3] [3] [5] Aprendizaje de fsck. [4]

[1] [2] [3] [3] [5] Aprendizaje de mount. [3]

[1] [2] [3] [3] [5] Aprendizaje de implementacion de funciones. [5]

[1] [2] [3] [3] [5] Aprendizaje de Diseño de Filesystem. [5]

Jorge Gatgens

[1] [2] [3] [3] [5] Aprendizaje de mkfs. [3]

[1] [2] [3] [3] [5] Aprendizaje de fsck. [4]

[1] [2] [3] [3] [5] Aprendizaje de mount. [3]

[1] [2] [3] [3] [5] Aprendizaje de implementacion de funciones. [5]

[1] [2] [3] [3] [5] Aprendizaje de Diseño de Filesystem. [5]

Estado Final del proyecto

Incompleto, se logro implementar casi que a su totalidad el file system, pero en memoria sin usar los qr's. También se logro la lectura y escriura de lectura de qr, junto con otras funciones necesarias para la integración de los qr al file system y una pila que manejara la fragmentación de estos, aunque es aquí donde se llego a los problemas y no se pudo terminar con la implementación.

Problemas encontrados y limitaciones

Primero, el siempre tan sonado problema con el manejo de la memoria en C, que aunque esta vez no fue tan grave por el uso de un debugger en condición, si represento en ocasiones perdida de tiempo valioso. Fuera de ahí, sí se comprendió en su totalidad como funcionaban las partes del file system y como implementarlos, pero planeación de tiempo sin considerar factores externos que nos pudieran afectar el tiempo efectivo para dedicarle al proyecto nos jugo uno mala pasada.

Lecciones Aprendidas

- Se aprendió a profundidad como funciona un file system desde el lado del programado.
- Manejo de qr en C.
- Un conocimiento mayor de Inodes.
- Importancia del superbloque y atributos de los archivos.
- Funcionamiento y utilidad de Fuse en general.

Bibliografía

- Hussain, M. (2019). Writing Less Simple, Yet Stupid Filesystem Using FUSE in C. Recuperado de: https://www.maastaar.net/fuse/linux/filesystem/c/2019/09/28/writing-less-simple-yet-stupid-filesystem-using-FUSE-in-C/
- Amay. (2015). Fuse_File_persistent_CS6233_NYU. Recuperado de: https://github.com/Amay22/Fuse_File_persistent_CS6233_NYU/blob/master/amayFuse.c
- Comunidad. (2020). LibFuse. Recuperado de: https://github.com/libfuse/libfuse
- Fukushi, K. (2020). libqrencode. Recuperado de: https://fukuchi.org/works/qrencode/