Sistema de Ficheros basado en Inodos PorygonExt

1A. Patricia Cárdenas Fernández
1B. David Gómez Delgado

***Abstract*— En este documento se muestra la descripción de la implementación de un sistema de ficheros simple basado en Inodos. Todo el código está en https://github.com/DavidGomezDelgado/FileSystem**

I. INTRODUCCIÓN

Este documento incluye información básica de la implementación de un Sistema de Ficheros simple basado en inodos e implementado con FUSE. La estructura de datos basada en Inodos nos permite un control mejor sobre la gestión de bloques en memoria, así como el formato ext2 como guía a la hora de implementar las funciones que aparecen a lo largo del documento.

II. OBJETIVOS

A continuación expondremos qué funcionalidades deberá presentar nuestra implementación:

A. Gestión de bloques

Es capaz de acceder y guardar información en bloques de memoria de manera organizada llenando los bloques desde el primero hasta el último, lo cual es posible gracias al uso de bitmaps (para recuento de bloques ocupados).
Lo que conocemos como bloques, son un conjunto de sectores del disco que almacenan información. Cada bloque, en nuestro caso, ocupa 4096 bytes, es decir, 4 KB.

B. Gestión de Inodos

El sistema de ficheros está basado en inodos, es decir, una estructura que guarda información variada sobre el fichero o directorio al que hace referencia. Gracias al bitmap de inodos, se puede llevar un registro de los inodos que hay en uso y mantener un orden a la hora de crearlos.

C. Gestión de ficheros

Se pueden crear, borrar o modificar ficheros accediendo a su correspondiente inodo. De esta forma, reúne todos los bloques de información que aguarda el inodo del fichero, de manera que, el usuario es capaz de leer, modificar o eliminar dicha información.

D. Gestión de directorios

La estructura de directorios tiene forma de árbol, debido a que todos los directorios cuelgan de uno, la raíz, "/". La estructura directorio también es un inodo pero los bloques asociados a éste guardan información sobre los archivos y directorios que contiene.
El sistema debe ser capaz de añadir ficheros al directorio, borrar elementos de él o borrar el directorio entero.

E. Estructurar de la memoria

La memoria ha de estructurarse de tal forma que sea compatible con el sistema que se ha creado. Por lo tanto, al principio se guardarán todos los metadatos del sistema de ficheros, es decir, Superbloque, inodos, bitmaps y, a continuación, todos los bloques de datos.

III. METODOLOGÍA

Para el proyecto hemos realizado una metodología ágil conocida como Scrum. Es decir, seguimos una metodología iterativa e incremental desde los puntos más básicos como podría ser guardar y cargar información en un bloque de memoria hasta completar un sistema de ficheros que recoja todos los objetivos ya mencionados. Para ello hemos realizado un seguimiento semanal, en el cual, para comenzar ponemos en común toda la información recogida, luego nos documentamos de las dudas que surgiesen para finalmente llevar a cabo la implementación de la sección escogida, así como asegurarnos de que el trabajo realizado anteriormente es totalmente compatible con la nueva implementación. Como añadido hemos seguido el método de programación pair-programming, que consiste en que un miembro programa mientras el otro corrige y busca documentación, haciendo que el código sea más robusto a medida que se va escribiendo.

IV. IMPLEMENTACIÓN

Nuestro sistema de ficheros se compone de un fichero formateado con toda la información que necesita FUSE para montar nuestro Sistema de ficheros además de la implementación de las funciones que modificarán el contenido del fichero que equivale al funcionamiento del Sistema de Ficheros. Todo el contenido ha sido mmapeado en el fichero para su posterior uso en la implementación de FUSE con un programa que deja formateado el fichero, es decir, que además se actualiza el bitmap de bloques con los bloques reservados y la creación del inodo raiz, este formateador está implementado en mkFilesystem.c. Los elementos que guardaremos en el fichero son los siguientes:

Superbloque

La información importante de nuestro sistema estará almacenada en una estructura superbloque, el cual se encuentra al principio de nuestro fichero formateado. La información que contiene es el Magic Number, el número de inodos así como el número de bloques, el tamaño de bloque y finalmente los offset que almacenan donde comienzan las

estructuras de los bitmap de bloques e inodos además de el inicio donde almacenamos todos los inodos y el inicio de donde almacenamos todos los bloques de datos.*Modificar si hay que diferenciar entre bloque de datos y bloque de entradas de directorio*

Magic number

En el superbloque necesitamos un identificador que nos asegure que el fichero que accedemos es el correcto. El magic number lo que permite es identificar que estamos abriendo los fichero en el formato correcto. Todos nuestros ficheros tienen el mismo formato, y por tanto, todos tendrán asociados dicho número identificativo. El valor que le hemos dado a esta variable es 123456

Tamaño de bloque

Para el tamaño de bloque hemos elegido 4K. En este caso, nos coincide con el tamaño de página, lo cual nos facilita el trabajo a la hora de mapear nuestro fichero.

Tamaño de fichero

Cada inodo va a guardar 10 punteros directos. Debido a ello, el tamaño que puede alcanzar cada fichero es el siguiente:

- 10 directos $\longrightarrow 4096 * 10 = 40960 = 40$ KB.
- 1 indirecto $\longrightarrow 40960 + \frac{4096}{8} * 4096 = 2138112 \approx 2$ GB.

Cada entrada de directorio está formada por un string que es el nombre del fichero y un puntero a su inodo. Esta estructura ocupa 32B donde 24B son para el nombre y 8 para el puntero. Por lo tanto, cada directorio podría guardar la siguiente cantidad de entradas:

- 10 directos $\longrightarrow \frac{4096}{64} * 10 = 64$ entradas de directorio.
- 1 indirecto $\longrightarrow 1280 + \frac{4096}{8} * \frac{4096}{64} = 34048$ entradas de directorio.

Número de inodos/bloques

Como nuestro sistema de ficheros esta orientado al almacenamiento, hemos decidido reservar una cuarta parte de todos los bloques reservados. De esta forma, nos quedamos con un mayor número de bloques de datos.

V. FUSE

El objetivo es el uso del fichero mmapeado en un sistema de ficheros implementado con FUSE. Para esto hemos decidido que nuestro sistema de ficheros como mínimo debe crear y eliminar tanto ficheros como directorios vacíos así como escritura y lectura de ficheros y renombrado de ficheros y directorios.

Funciones de FUSE

Están definida en una estructura, dicha estructura contiene para su correcto funcionamiento getattr, readdir, open, read, write, rmdir, mkdir, create, unlink, rename, utimens y truncate.Todas las funciones están en el fichero FilesystemFUSE.

VI. FUTURAS MEJORAS

Para un futuro desarrollo del Sistema de Ficheros con el fin de conseguir mejorar tenemos en cuenta implementar:

Hard Links y Soft Links Implementación para tener un sistema de ficheros mucho más completo. **Punteros indirectos** Estos nos permitiran si es un fichero almacenar más contenido en estos, en caso de ser un directorio podremos tener más entradas.

Fecha de creación Actualmente la fecha de creación de cualquier fichero y directorio es la misma fecha de nuestro fichero formateado, para un futuro nos gustaría poder guardar la fecha real de creación de cada inodo con la finalidad de tener un mayor control sobre estos.

Mejora de eficencia La búsqueda de ficheros y directorios está hecho de manera recursiva, para mejorar la eficiencia nos gustaría cambiar la estructura a un Árbol de Búsqueda Rojo-Negro.

Desfragmentación Nuestro Sistema de Ficheros no tiene un control sobre la posible fragmentación interna o externa, en el futuro podremos implementar una eficencia en nuestro formateo del fichero para que al almacenar todo el contenido no contenga prácticamente ningún tipo de fragmentación mencionado. **Encriptación** Para un uso más exclusivo y personal implementaremos un sistema de encriptado para que el usuario pueda elegir quien quiere que vea el contenido de su sistema de ficheros.

VII. CONCLUSIONES

Todo el trabajo realizado en el desarrollo del sistema de ficheros nos ha permitido alcanzar una mayor compresión de su funcionamiento, como por ejemplo el manejo de los bloques, cómo se mapean los ficheros, así como formatear un fichero para poder usarlo en un sistema de ficheros implementado con FUSE.

Todo este esfuerzo nos permitirá seguir avanzando con el proyecto en un futuro e implementar las mejoras planteadas en el apartado anterior.

VIII. HERRAMIENTAS

A. Editores de Texto

- *Neovim*: Editor de código de terminal.
- *Visual Studio Code*: Editor de código con gran cantidad de extensiones y que permite realizar el debugging del código.
- *Overleaf*: Editor de documentos LaTeXen línea que permite trabajar de forma paralela en el mismo proyecto.
- *Geany*: Editor de código muy ligero.

B. Diagramas Visuales

- LucidChart: Elaboración de diagramas de ejecución

C. Compilación y Debugging

- *Gcc*: Compilador de lenguaje C
- *Gdb*: Depurador de código en C

REFERENCES

[1] S. Romero, "Diseño de Sistemas Operativos" Información en Campus Virtual Uma, tutorías y clases.
[2] https://libfuse.github.io/doxygen/structfuse__operations.html
[3] Pedro, "¿Qué es sistema de archivos Ext2/Ext3/Ext4 y Cuál es la diferencia? [Guía completa]" https://es.easeus.com/partition-manager-tips/sistema-de-archivos-ext2-ext3-ext4.html.
[4] Mª Asunción, "Tema 5. Estructura e Implementación del sistema de ficheros", Universidad de Jaume I
[5] "Tema 4. Sistema de archivos", Departamento de Lenguajes y Computación, Universidad de Almería
[6] https://manpages.ubuntu.com/
[7] https://man7.org/linux/man-pages/
[8] https://www.ibm.com/docs/en/zos/2.3.0?topic=functions-munmap-unmap-pages-memory