VIGILANCIA EN UN SISTEMA DE ARCHIVOS

Laboratorio 4
Sistemas Operativos 2022
FaMAFyC - UNC

¿DE QUÉ ESTAMOS HABLANDO?

Un sistema de archivos es un componente del sistema operativo que se encarga de **administrar** y **organizar** los datos dentro de un dispositivo de memoria.

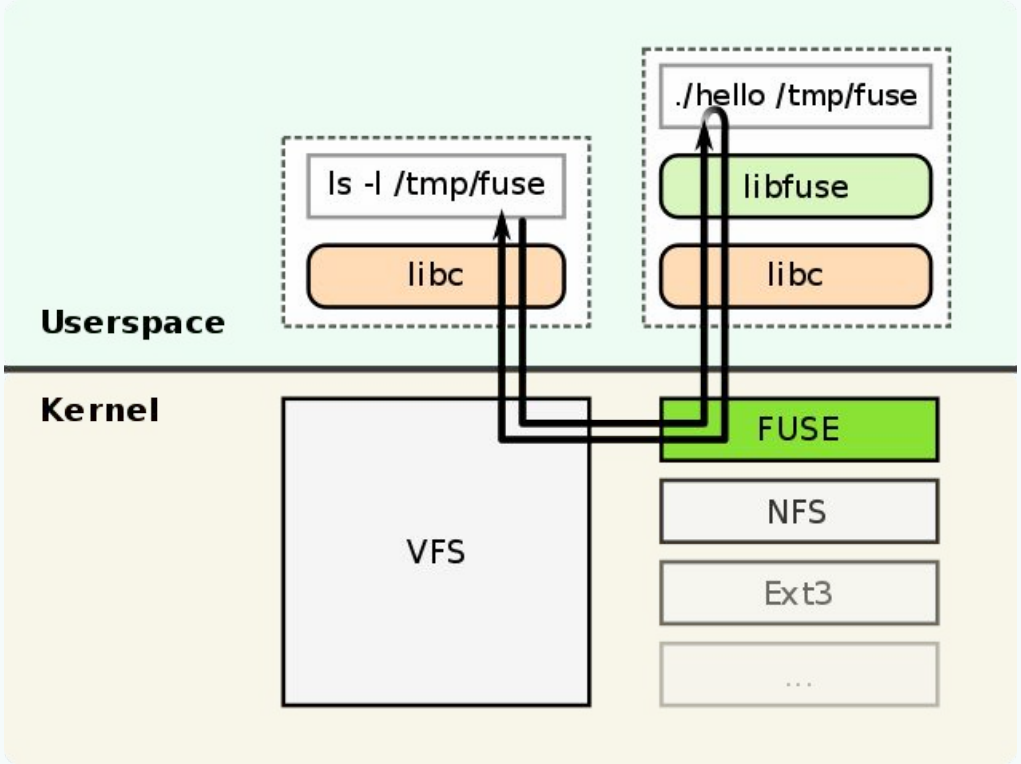
En nuestro caso, organizaremos los datos en archivos y directorios, que tienen metadatos y una jerarquía definida por el usuario

CONSIGNAS

Lo que tienen que hacer para aprobar

01

FILE SYSTEM IN USER SPACE (FUse)



IMPLEMENTACIÓN

01 Registrar la actividad del usuario

Crear un archivo en donde se guardarán registros de las actividades del usuario.

02 Esconder el registro

Modificar el sistema para que el archivo con los logs de actividad sean invisibles para el usuario tanto si la imagen es montada con fat-fuse o con otra herramienta.

03 Borrar archivos y directorios

Implementar las operaciones unlink y rmdir de manera que se puedan borrar archivos y directorios del FS

OBJETIVOS DE APRENDIZAJE

- Reconocer los distintos conceptos y niveles de abstracción del sistema de archivos vistos en el teórico, y hacer un paralelismo con la implementación de FAT con FUSE.

OBJETIVOS DE APRENDIZAJE

- Reconocer los distintos conceptos y niveles de abstracción del sistema de archivos vistos en el teórico, y hacer un paralelismo con la implementación de FAT con FUSE.
- Identificar las partes relevantes a un problema en una base de código extensa y complicada.

OBJETIVOS DE APRENDIZAJE

- Reconocer los distintos conceptos y niveles de abstracción del sistema de archivos vistos en el teórico, y hacer un paralelismo con la implementación de FAT con FUSE.
- Identificar las partes relevantes a un problema en una base de código extensa y complicada.
- Practicar (más) programación

OBJETIVOS DE APRENDIZAJE

- Reconocer los distintos conceptos y niveles de abstracción del sistema de archivos vistos en el teórico, y hacer un paralelismo con la implementación de FAT con FUSE.
- Identificar las partes relevantes a un problema en una base de código extensa y complicada.
- Practicar (más) programación
- Horrorizarse por lo fácil que es implementar un sistema de vigilancia transparente para el usuario y poner en perspectiva la importancia de los sistemas operativos de código abierto.

ENTREGA

Requisitos indispensables

02

¡OTRA VEZ ARROZ CON POLLO!

REPOSITORIO

A través de Bitbucket, con commits pequeños, con nombres significativos, y de todos los integrantes

ESTILO

Tienen que usar la herramienta clang-format para asegurar que el estilo de código sea consistente con el entregado. Se integra fácilmente con git.

TESTS

El código debe pasar los tests provistos por la cátedra. Sin embargo, eso no es garantía de que no tengan errores

FECHA DE ENTREGA

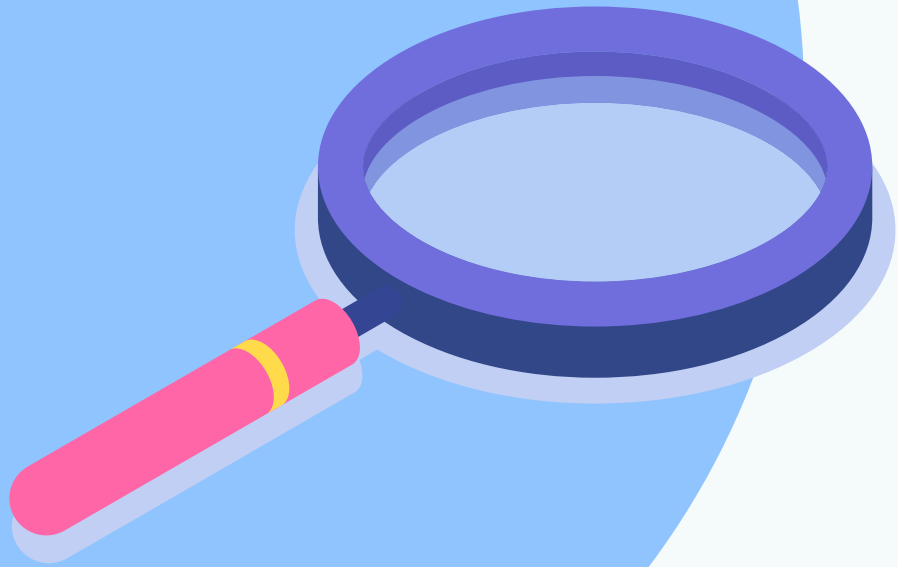
15/11/2022

15:59:00.000

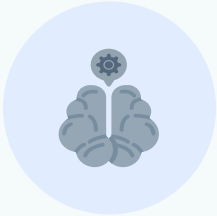
03

INTRO A FAT32

¿Cómo se guardan los datos en un pen-drive?



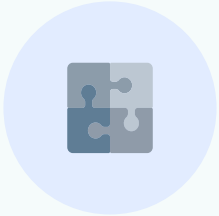
LOS COMPONENTES DE FAT



SECTOR DE BOOT

Define cómo está organizado el volúmen. Ej:

- Particiones
- ubicación de la **FAT**
- segmentos del sector de datos **(clusters)**



TABLA(S) FAT

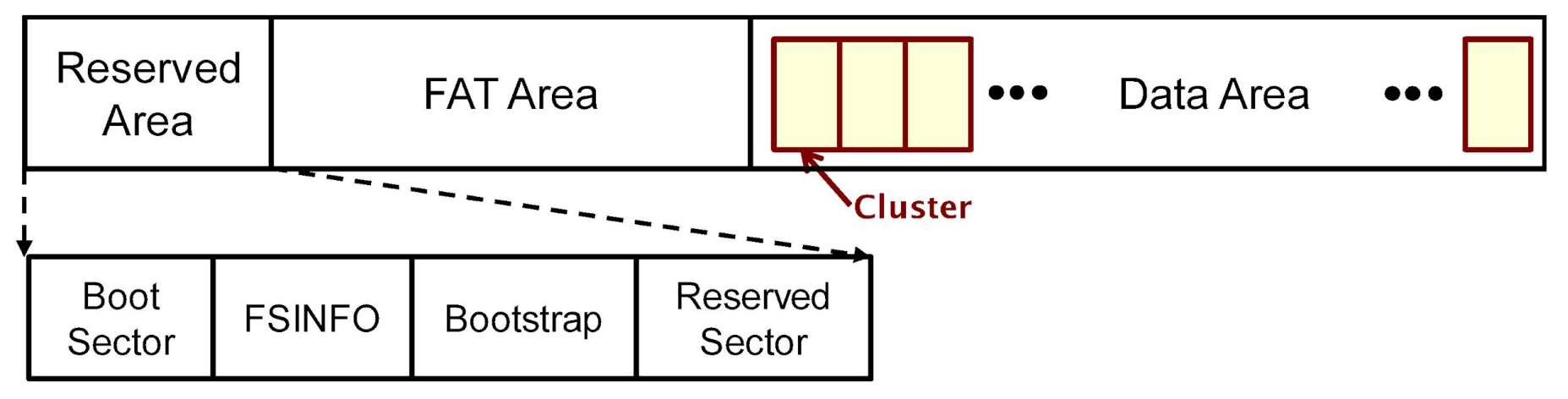
Define dónde están los datos cada archivo.
Como los datos pueden no estar el clusters contiguos, la FAT guarda la cadena de clusters del archivo.

SECTOR DE DATOS

Está dividido en clusters contiguos del mismo tamaño

- Clusters de datos
- Clusters de entradas de directorio

LOS COMPONENTES DE FAT



ENTRADAS DE DIRECTORIO

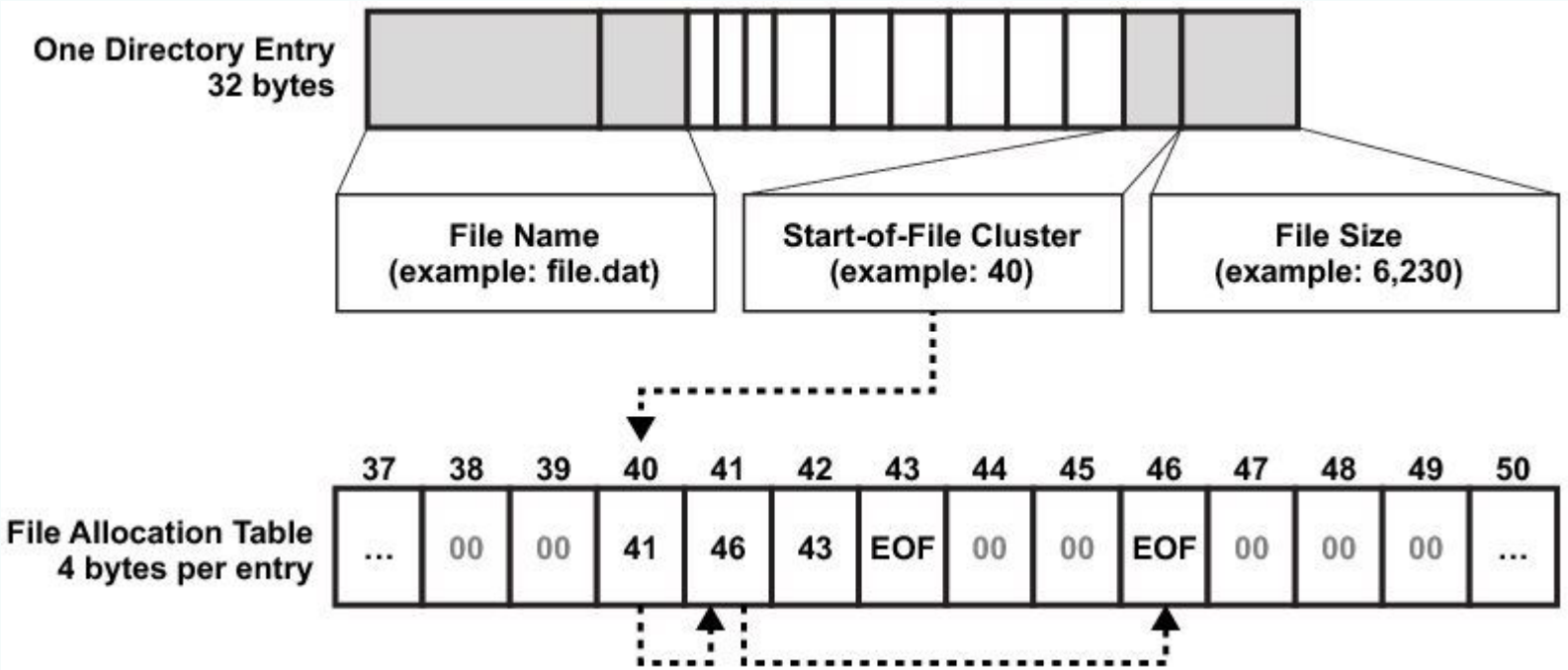
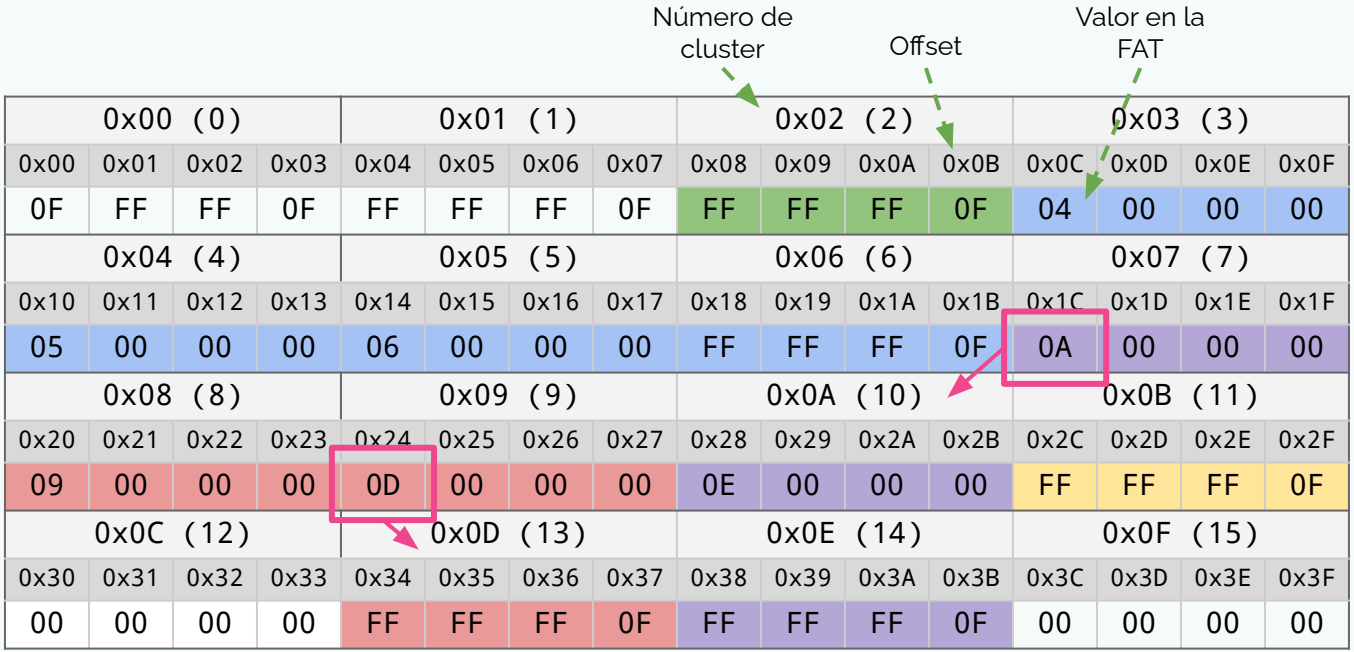


TABLA FAT32



0x00 (0)				0x01 (1)				0x02 (2)				0x03 (3)			
0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0F	FF	FF	0F	FF	FF	FF	0F	FF	FF	FF	0F	04	00	00	00
0x04 (4)				0x05 (5)				0x06 (6)				0x07 (7)			
0x10	0x11	0x12	0x13	0x14	0x15	0x16	0x17	0x18	0x19	0x1A	0x1B	0x1C	0x1D	0x1E	0x1F
05	00	00	00	06	00	00	00	FF	FF	FF	0F	0A	00	00	00
0x08 (8)				0x09 (9)				0x0A (10)				0x0B (11)			
0x20	0x21	0x22	0x23	0x24	0x25	0x26	0x27	0x28	0x29	0x2A	0x2B	0x2C	0x2D	0x2E	0x2F
09	00	00	00	0D	00	00	00	0E	00	00	00	FF	FF	FF	0F
0x0C (12)				0x0D (13)				0x0E (14)				0x0F (15)			
0x30	0x31	0x32	0x33	0x34	0x35	0x36	0x37	0x38	0x39	0x3A	0x3B	0x3C	0x3D	0x3E	0x3F
00	00	00	00	FF	FF	FF	0F	FF	FF	FF	0F	00	00	00	00

Offset 600
Size 2000

- Entrada del directorio raíz. Primer cluster de la cadena
- Cadena correspondiente al primer archivo - Todos los clusters son contiguos
- Cadena correspondiente al tercer archivo - Los clusters están divididos en tres porciones no contiguas
- Cadena correspondiente al quinto archivo - Tiene un solo cluster
- Clusters libres