

Taller Ext2

Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina


Sistemas Operativos, primer cuatrimestre de 20202 - virtual

(2) Presentación Taller

- Se programará un Sistema de Archivos ext2.
- Puntualmente, se requiere leer el contenido de `/grupos/gNUMERO/nota.txt`.
- ¿Qué debemos conocer y tener para lograrlo?
 - Lo que aprendimos en la teórica y la clase complementaria sobre ext2.
 - Un disco al cual podemos acceder a cualquiera de sus bloques.

(3) El disco

- ¿Qué es? Un montón de bytes agrupados en bloques.
- A cada bloque se accede con su LBA (Logical Block Addressing).
- API de HDD:

```
int read(unsigned int lba, unsigned char * buffer);  
int write(unsigned int lba, unsigned char * buffer);
```
- ¿Qué tamaño tiene el disco? A priori no se conoce.
- ¿Qué tamaño tiene cada bloque? 512 bytes.
- ¿Por dónde se empieza? 

(4) Boot block


- *Bloque de Booteo o Master Boot Record*
- Está en la primera parte del disco. Es el espacio reservado ya mencionado de 1024 bytes.

Structure of a classical generic MBR

Address		Description	Size (bytes)
Hex	Dec		
+000h	+0	Bootstrap code area	446
+1BEh	+446	Partition entry #1	16
+1CEh	+462	Partition entry #2	16
+1DEh	+478	Partition entry #3	16
+1EEh	+494	Partition entry #4	16
+1FEh	+510	55h	2
+1FFh	+511	AAh	
Total size: 446 + 4×16 + 2			512

- No contiene datos en TODOS los sistemas de archivos. Solo los usados para iniciar la máquina.

(5) Partición de EXT2

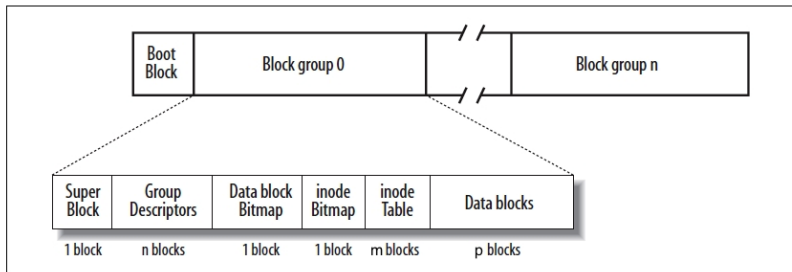
- Llegamos hasta donde empieza el primer grupo de bloques.
- El superblock: tiene información de TODO el sistema de archivos.
- ¿En qué parte de la partición estará?
- A partir del byte 1024. Independientemente, del tamaño del bloque. 

(6) Superblock

```
struct Ext2FSSuperblock {
__le32 s_inodes_count; /* Contador de Inodos */
__le32 s_blocks_count; /* Contador de Bloques */
__le32 s_r_blocks_count; /* Contador de Bloques reservados */
__le32 s_free_blocks_count; /* Contador de Bloques libres */
__le32 s_free_inodes_count; /* Contador de Inodos libres */
__le32 s_first_data_block; /* Primer bloque de Datos */
__le32 s_log_block_size; /* Tamano del bloque */
...
__le32 s_blocks_per_group; /* Cantidad de bloques por grupo */
...
__le32 s_inodes_per_group; /* Cantidad de Inodos por grupos */
...
__le16 s_magic; /* Firma magica – Identifica el S.A. */
__le32 s_first_ino; /* Primer Inodo no reservado */
__le16 s_inode_size; /* Tamano de la estructura del Inodo */
```

(7) Estructura de Ext2

- ¿Dónde está mi archivo /home/krypton.gis ?
- Recordemos que en Ext2 todo está representado por Inodos.
¿Cuál es el inodo de mi archivo?
- Supongamos que está en el Inodo 2483.



(8) Inodo

- La representación de un archivo.
- Un archivo puede ser desde un archivo regular, un directorio, un pipe, un socket, un device, etc.
- A bajo nivel, en este taller, es un struct de FSInode.

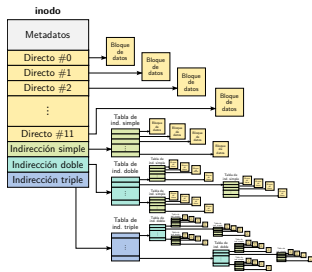
(9) FSInode

```
struct Ext2FSInode {  
    unsigned short mode;  
    unsigned short uid;  
    unsigned int size;  
    unsigned int atime;  
    unsigned int ctime;  
    unsigned int mtime;  
    unsigned int dtime;  
    unsigned short gid;  
    unsigned short links_count;  
    unsigned int blocks;  
    unsigned int flags;  
    unsigned int os_dependant_1;  
    unsigned int block[15];  
    unsigned int generation;  
    unsigned int file_acl;  
    unsigned int directory_acl;  
    unsigned int faddr;  
    unsigned int os_dependant_2[3];  
}
```

- ¿Dónde están los datos? ⚠
- ¿Dónde está el nombre del archivo? ⚠ A nivel de usuario no se hace referencia a números de Inodos.
- ¿El Inodo directorio qué struct usa? ⚠


(10) Inodo - Datos

- 15 Punteros a bloques con distintos sabores:
 - 12 Punteros a bloques de datos directos
 - 1 Puntero indirecto a bloque de datos
 - 1 Puntero con una doble indirección a bloque de datos
 - 1 Puntero con una triple indirección a bloque de datos.




- ¿Son punteros a direcciones de memoria? ⚠
- ¿Los bloques a los que apuntan, están en memoria o en disco? ⚠

(11) Inodo - Directorio

- Es un Inodo IGUAL que cualquier otro.
- Es decir, tiene la misma estructura Ext2FSInode.
- Entonces ¿Dónde está la lista de archivos de mi directorio?
- En los bloques de datos. 

(12) Ext2FSInode

```
struct Ext2FSDirEntry {  
    unsigned int inode;  
    unsigned short record_length;  
    unsigned char name_length;  
    unsigned char file_type;  
    char name[];  
};
```

- **Los datos** del Inodo son una lista de structs Ext2FSDirEntry.
- Cada struct tiene tamaño variable. 


(13) Enunciado

- Completar la implementación de los siguientes métodos:
 - `unsigned int get_block_address(inode,block_number)`
 - `Ext2FSInode * load_inode(inode_number)`
 - `Ext2FSInode * get_file_inode_from_dir_inode(from,filename)`
- Hacer un programa que, utilizando el FS programado en el punto anterior, imprima los 50 caracteres que se encuentran guardados en el archivo `/grupos/gNUMERO/nota.txt` (de la imagen de disco `hdd.raw` provista) a partir de la posición 14000 inclusive.

(14) ¿Qué parte del código ya está preparada?

- Las clases HDD, MBR y PartitionEntry.
- Parcialmente la clase Ext2FS.
- Las estructuras de Ext2FS:
 - Ext2FSSuperblock (Superblock)
 - Ext2FSBlockGroupDescriptor (Block Group Descriptor)
 - Ext2FSInode (Inode)
 - Ext2FSDirEntry (Directory Entry)
- Las funciones auxiliares de Ext2FS:
 - read_block: Lee un bloque de disco.
 - superblock: Devuelve el superbloque.
 - block_group: Devuelve el descriptor del bloque de grupo.
 - blockgroup_for_inode: Número de blockgroup del Inodo.
 - blockgroup_inode_index: Offset dentro de la tabla de Inodos, para el inodo.

(15) Últimos tips

- Hagan los ejercicios en el orden dado.
- Descompriman la imagen hdd.raw.gz en /tmp para usarla.
- Hay estructuras para cada tipo.
- Utilicen las funciones auxiliares.
- Recuerden, los directorios son archivos. 
- Documentación:
 - <http://www.nongnu.org/ext2-doc/ext2.html>
 - <http://e2fsprogs.sourceforge.net/ext2intro.html>
 - <http://wiki.osdev.org/Ext2>