

Taller Ext2

Sergio Romano

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


Sistemas Operativos, primer cuatrimestre de 2015

(2) Presentación Taller

- Hoy vamos a programar ext2
- ¿Qué tenemos para hacerlo?
 - Lo que aprendimos en la teoría sobre ext2
 - Lo que aprendimos en la práctica sobre ext2
 - Un disco al cual podemos acceder a cualquiera de sus bloques
- ¿Dudas?
- **A programar entonces!**

(3) El disco

- ¿Qué es? Un montón de bits agrupados en bloques.
- A cada bloque lo accedo 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? Ni idea
- ¿Qué tamaño tiene cada bloque? 512 bytes (ver Advanced Format)
- ¿Por dónde empiezo? 

(4) MRB


- Master Boot Record
- El primer bloque del disco

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 está en TODOS los discos.
- GPT busca jubilarlo. (32bits para indicar la LBA de cada partición)

(5) Partición de EXT2

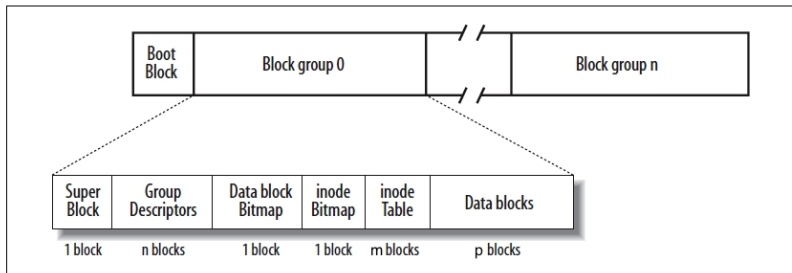
- Llegamos hasta donde empieza ext2. ¿Y ahora?
- El superblock. El que tiene **la posta**.
- ¿En qué bloque de la partición estará?
- Exacto, en el tercer bloque. WTF?
- En realidad, siempre en el byte 1024. Independientemente, del tamaño del bloque. 

(6) Superblock

```
struct Ext2FSSuperblock {
__le32 s_inodes_count; /* Inodes count */
__le32 s_blocks_count; /* Blocks count */
__le32 s_r_blocks_count; /* Reserved blocks count */
__le32 s_free_blocks_count; /* Free blocks count */
__le32 s_free_inodes_count; /* Free inodes count */
__le32 s_first_data_block; /* First Data Block */
__le32 s_log_block_size; /* Block size */
...
__le32 s_blocks_per_group; /* # Blocks per group */
...
__le32 s_inodes_per_group; /* # Inodes per group */
...
__le16 s_magic; /* Magic signature */
__le32 s_first_ino; /* First non-reserved inode */
__le16 s_inode_size; /* size of inode structure */
```

(7) Estructura de Ext2

- Todo muy lindo pero ¿dónde está mi archivo /home/krypton.gis ?






- La bola mágica me dijo que está en el inodo 2483.

(8) Inodo

- La representación de un archivo
- Un archivo puede ser desde un archivo regular, hasta un directorio, un pipe, un socket, un device, etc.
- Hoy, para nosotros, una struct de FSInode

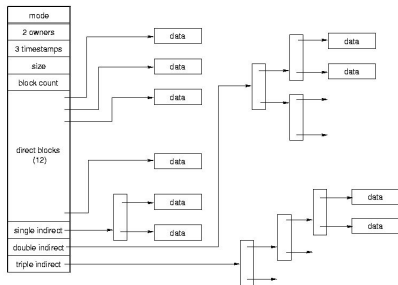
(9) FSInode

```
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?  Porque la gente no anda preguntando por números de inodos.
- ¿El inodo directorio qué tipo tiene? 


(10) Inodo - Datos

- 15 Punteros a bloques con trucos. IDDQD
 - 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.




- ¿Por qué hicieron este quilombo? ⚠
- ¿En qué parte del disco están cada bloque? ⚠

(11) Inodo - Directorio

- Es un inodo IGUAL que cualquier otro.
- Es decir, tiene la misma estructura Ext2FSInode.
- Entonces? Dónde están los archivos de mi directorio?
- En los bloques de datos.
- Repito, 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 un arreglo de struct Ext2FSDirEntry
- La struct tiene tamaño variable. 
- ¿Cómo saber cuantas structs tengo en mi arreglo? Para pensar.
- ¿De verdad vas a usar un arreglo si tienen tamaño variable? Apa-la-la



(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 17 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 tengo solucionado?

- Clases HDD, MBR y PartitionEntry resueltas
- Clase Ext2FS parcialmente
- Estructuras de Ext2FS
 - Ext2FSSuperblock (Superblock)
 - Ext2FSBlockGroupDescriptor (Block Group Descriptor)
 - Ext2FSInode (Inode)
 - Ext2FSDirEntry (Directory Entry)
- 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
- Los directorios son archivos 
- ¿Los directorios son archivos?
- Sí, los directorios son archivos 
- Documentación
 - <http://www.nongnu.org/ext2-doc/ext2.html>
 - <http://e2fsprogs.sourceforge.net/ext2intro.html>
 - <http://wiki.osdev.org/Ext2>
 - <http://oreilly.com/catalog/linuxkernel2/chapter/ch17.pdf>