

# Sistemas operativos y Redes II

Trabajo Práctico

Profesor: Alexis Tcach

Fecha de entrega: Jueves 03 de Mayo de 2018

Integrantes:

Dario J. Rick

Nicolás J. Cabral

## Ejercicios:

### 1. Al montarlo. > Para que se ha puesto umask=000 ?

Se ha puesto umask 000 para que los directorios de la imagen montada tengan permiso de lectura/escritura/ejecución para todos (permiso 777). De lo contrario solo se podría acceder via sudo.

### 2. Cargando el MBR

a) Muestre el MBR con el Hex Editor. Muestre los primeros bytes y la tabla de particiones. ¿Cuántas particiones hay ? Muestre claramente en qué lugar puede observarlo.

La estructura del MBR define que es posible almacenar hasta 4 particiones primarias

### MBR Data Structure

Byte Range	Description	Essential
0-445	Boot Code	No
446-461	Partition Table Entry #1	Yes
462-477	Partition Table Entry #2	Yes
478-493	Partition Table Entry #3	Yes
494-509	Partition Table Entry #4	Yes
510-511	Signature value (0xAA55)	Yes

Si vemos en un editor hexadecimal los bytes 446 al 461 de la imagen, notamos que tenemos declarada una partición primaria

HxD - [C:\Users\dario\Documents\Repos\SOR2\_TP1\entregable\test.img]

```

Archivo Edición Buscar Ver Análisis Extras Ventanas ?
16 ANSI hex
test.img

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 EB 3C 90 6D 6B 66 73 2E 66 61 74 00 02 04 01 00  è<.mkfs.fat.....
00000010 02 00 02 00 08 F8 02 00 20 00 40 00 00 00 00 00  ....ø... .@.....
00000020 00 00 00 00 80 00 29 5F 05 C8 06 4E 4F 20 4E 41  ....€.)_.È.NO NA
00000030 4D 45 20 20 20 20 46 41 54 31 32 20 20 20 0E 1F  ME   FAT12   ..
00000040 BE 5B 7C AC 22 C0 74 0B 56 B4 0E BB 07 00 CD 10  %[|~"Àt.V'.»...í.
00000050 5E EB F0 32 E4 CD 16 CD 19 EB FE 54 68 69 73 20  ^è&2áí.í.épThis
00000060 69 73 20 6E 6F 74 20 61 20 62 6F 6F 74 61 62 6C  is not a bootabl
00000070 65 20 64 69 73 6B 2E 20 20 50 6C 65 61 73 65 20  e disk. Please
00000080 69 6E 73 65 72 74 20 61 20 62 6F 6F 74 61 62 6C  insert a bootabl
00000090 65 20 66 6C 6F 70 70 79 20 61 6E 64 0D 0A 70 72  e floppy and..pr
000000A0 65 73 73 20 61 6E 79 20 6B 65 79 20 74 6F 20 74  ess any key to t
000000B0 72 79 20 61 67 61 69 6E 20 2E 2E 2E 20 0D 0A 00  ry again ... ..
000000C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000000D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000000E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000000F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000180 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
00000190 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000001A0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000001B0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000001C0 02 00 01 20 20 00 01 00 00 00 FF 07 00 00 00 00  ... ..ÿ...
000001D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000001E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ....
000001F0 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA  ....U*
00000200 F8 FF FF 00 F0 FF 00 F0 FF 00 00 00 00 00 00 00  øÿÿ.ÿÿ.ÿÿ.....

```

El resto de los bytes están en 0, por lo que podemos considerar que es la única partición.

b) Lea los datos del punto anterior y muestrelos por pantalla mediante por código C.

Código en el archivo "read\_mbr.c"

```
nicolas@nicolas-ubuntu:~/Escritorio/SOR2_TP1/entregable$ ./read_mbr
Partition entry 0: First byte 80
  Partition start in CHS: 00:02:00
  Partition type 01
  FAT12
  Partition end in CHS: 00:20:20
  start_sector 1
  length_sectors 2047
  Relative LBA address 0x00000001, 2047 sectors long
Partition entry 1: First byte 00
  Partition start in CHS: 00:00:00
  Partition type 00
  Tipo de particion no reconocida
  Partition end in CHS: 00:00:00
  start_sector 0
  length_sectors 0
  Relative LBA address 0x00000000, 0 sectors long
Partition entry 2: First byte 00
  Partition start in CHS: 00:00:00
  Partition type 00
  Tipo de particion no reconocida
  Partition end in CHS: 00:00:00
  start_sector 0
  length_sectors 0
  Relative LBA address 0x00000000, 0 sectors long
Partition entry 3: First byte 00
  Partition start in CHS: 00:00:00
  Partition type 00
  Tipo de particion no reconocida
  Partition end in CHS: 00:00:00
  start_sector 0
  length_sectors 0
  Relative LBA address 0x00000000, 0 sectors long
nicolas@nicolas-ubuntu:~/Escritorio/SOR2_TP1/entregable$
```

c) Muestre en el Hex Editor si la primer particion es booteable o no. ¿Lo es?

Vemos en la documentación que el boot flag se encuentra en el primer byte

Byte Range	Description	Essential
0-0	Bootable Flag	No
1-3	Starting CHS Address	Yes
4-4	Partition Type	Yes
5-7	Ending CHS Address	Yes
8-11	Starting LBA Address	Yes
12-15	Size in Sectors	Yes

Por tanto la primer particion es booteable, dado que tiene el primer byte 80.

test.img

00000000 EB 3C 90 6D 6B 66 73 2E 66 61 74 00 02 04 01 00 02 00 02 00 08 F8 02 00 20 00 40 00 00 00 <.mkfs.fat.....@...

00000001 00 00 00 00 00 00 80 00 29 5F 05 C8 06 4E 4F 20 4E 41 4D 45 20 20 20 20 46 41 54 31 32 20 .....)\_...NO NAME FAT12

00000003 20 20 0E 1F BE 5B 7C AC 22 C0 74 0B 56 B4 0E BB 07 00 CD 10 5E EB F0 32 E4 CD 16 CD 19 EB ...[|.t.V.....^..2.....

00000005 FE 54 68 69 73 20 69 73 20 6E 6F 74 20 61 20 62 6F 6F 74 61 62 6C 65 20 64 69 73 6B 2E 20 .This is not a bootable disk.

00000007 20 50 6C 65 61 73 65 20 69 6E 73 65 72 74 20 61 20 62 6F 6F 74 61 62 6C 65 20 66 6C 6F 70 Please insert a bootable flop

00000009 70 79 20 61 6E 64 0D 0A 70 72 65 73 73 20 61 6E 79 20 6B 65 79 20 74 6F 20 74 72 79 20 61 py and..press any key to try a

0000000b 67 61 69 6E 20 2E 2E 2E 20 0D 0A 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 gain ... ..

0000000d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0000000f 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000011 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000013 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000015 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000017 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000019 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0000001b 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0000001d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

0000001f 55 AA F8 FF FF 00 F0 FF 00 F0 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 U.....

00000021 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

00000023 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Signed 8 bit: 0

Unsigned 8 bit: 0

Signed 16 bit: 2

Unsigned 16 bit: 2

☐ Show little endian decoding

Signed 32 bit: 131073

Unsigned 32 bit: 131073

Float 32 bit: 1.836724E-40

Float 64 bit: 2.78136620594796E-309

☐ Show unsigned as hexadecimal

Hexadecimal: 00 02 00 01

Decimal: 000 002 000 001

Octal: 000 002 000 001

Binary: 00000000 00000010 00000000 00000001

ASCII Text:

Offset: 0x1bf / 0xffff Selection: 0x1be to 0x1be (0x1 bytes) INS

d) Muestre, mediante un programa en C, para la primer particion: el flag de booteable, la dirección Cylinder-head-sector (chs), el tipo de partición y su tamaño en sectores.

Código en el archivo `read_mbr.c` donde imprime la los datos de la primer particion.

Datos de la primera partición:

Flag Booteable: Primer Byte 80

CHS:

Partición Start en CHS: 00:02:00



Partición End en CHS: 00:20:20

TIPO DE PARTICIÓN: FAT12

SECTORES: Tamaño 2047

```
root@dario-VirtualBox:/home/dario/Repos/SOR2_TP1/entregable# ./read_mbr
Partition entry 0: First byte 80
  Partition start in CHS: 00:02:00
  Partition type 01
  FAT12
  Partition end in CHS: 00:20:20
  start_sector 1
  length_sectors 2047
  Relative LBA address 0x00000001, 2047 sectors long
Partition entry 1: First byte 00
  Partition start in CHS: 00:00:00
  Partition type 00
  Tipo de particion no reconocida
  Partition end in CHS: 00:00:00
  start_sector 0
  length_sectors 0
  Relative LBA address 0x00000000, 0 sectors long
Partition entry 2: First byte 00
  Partition start in CHS: 00:00:00
  Partition type 00
  Tipo de particion no reconocida
  Partition end in CHS: 00:00:00
  start_sector 0
  length_sectors 0
  Relative LBA address 0x00000000, 0 sectors long
Partition entry 3: First byte 00
  Partition start in CHS: 00:00:00
  Partition type 00
  Tipo de particion no reconocida
  Partition end in CHS: 00:00:00
  start_sector 0
  length_sectors 0
  Relative LBA address 0x00000000, 0 sectors long
```

### 3. Cargando la tabla de archivos

a) ¿Cuántos y cuáles archivos tiene el filesystem? Muestrelos con Bless y genere el código para mostrarlos.

Si montamos la imagen, vemos que en el directorio principal existe el archivo "hola.txt" y el directorio "mi\_dir", el cual contiene internamente un archivo llamado "vacío.txt"

```

Escritura Mount >>help para obtener mas informacion.
root@dario-VirtualBox:/home/dario/Repos/SOR2_TP1/entregable# mount test.img /mnt/img -t msdos -o loop
root@dario-VirtualBox:/home/dario/Repos/SOR2_TP1/entregable# cd /mnt/img
root@dario-VirtualBox:/mnt/img# ls
hola.txt  mi_dir
root@dario-VirtualBox:/mnt/img# cd mi_dir/
root@dario-VirtualBox:/mnt/img/mi_dir# ls
vacio.txt
root@dario-VirtualBox:/mnt/img/mi_dir#

```

```

nicolas@nicolas-ubuntu:~/Escritorio/SOR2_TP1/entregable$ ./read_root
Encontrada particion FAT12 0
En 0x200, sector size 512, FAT size 2 sectors, 2 FATs

Sectorres reservados: 1
Tamaño de sector: 512
Cantidad tablas FAT: 2
Tamaño de FAT: 2

Inicio Root directory en 0xA00
Root dir_entries 512
Posición: [0x0] File: [Am.]
Posición: [0x1] File: [MI_DIR . ]
Posición: [0x2] File: [Ah.]
Posición: [0x3] File: [HOLA .TXT]
Deleted file: [?..]
Deleted file: [?..]
Deleted file: [?ORRAR~1.SWP]
Deleted file: [?..]
Deleted file: [?..]
Deleted file: [?ORRAR~1.SWX]

Leído Root directory, ahora en 0x4A00
nicolas@nicolas-ubuntu:~/Escritorio/SOR2_TP1/entregable$

```

El FileSystem tiene 2 archivos .

MI\_DIR - HOLA.TXT

El código se encuentra en el archivo *read\_boot*.

**b) ¿Hay archivos borrados ? ¿Cuales ?**

Si. En el programa “read\_root.c” se ve que salen otros archivos además, pero los mismos se encuentran borrados.

**c) ¿Qué puede decir acerca del recupero de archivos ?**

Si se les cambia el primer byte desde 0xE5 a 0x05, se debería poder recuperar el archivo

```
void print_file_info(FILE* in, Fat12Entry *entry, int tamanoEntry, Fat12BootSector *bs, int posicion)
{
    switch (entry->filename[0]) {
        case 0x00:
            return; // unused entry
        case 0xE5:
            printf("\n-----\n");
            printf("Deleted file: [?%.7s.%.3s] ", entry->filename + 1, entry->ext);
            break;
        case 0x05:
            printf("\n-----\n");
            printf("File starting with 0xE5: [%.7s.%.3s] ", 0xE5, entry->filename + 1, entry->ext);
            break;
        case 0x2E:
            printf("\n-----\n");
            printf("Directory: [%.8s.%.3s] ", entry->filename, entry->ext);
            break;
        default: //Si cae en este caso, es el primer caracter del archivo
            printf("\n-----\n");
            printf("File: [%.8s.%.3s] \n", entry->filename, entry->ext);
            printf("Cluster de inicio [0x%X] \n", entry->cluster_inicio); //tengo que sumar offset
            //First allocation unit empieza en 0x5800
            printf("Tamaño de archivo [%i] bytes \n", (int)entry->tamano_archivo[0]);
    }
}
```

Esto se puede hacer siempre y cuando no se cree un nuevo archivo que apunte al mismo cluster, ya que siendo así, el mismo se pisaría y no habría forma de recuperarlo.

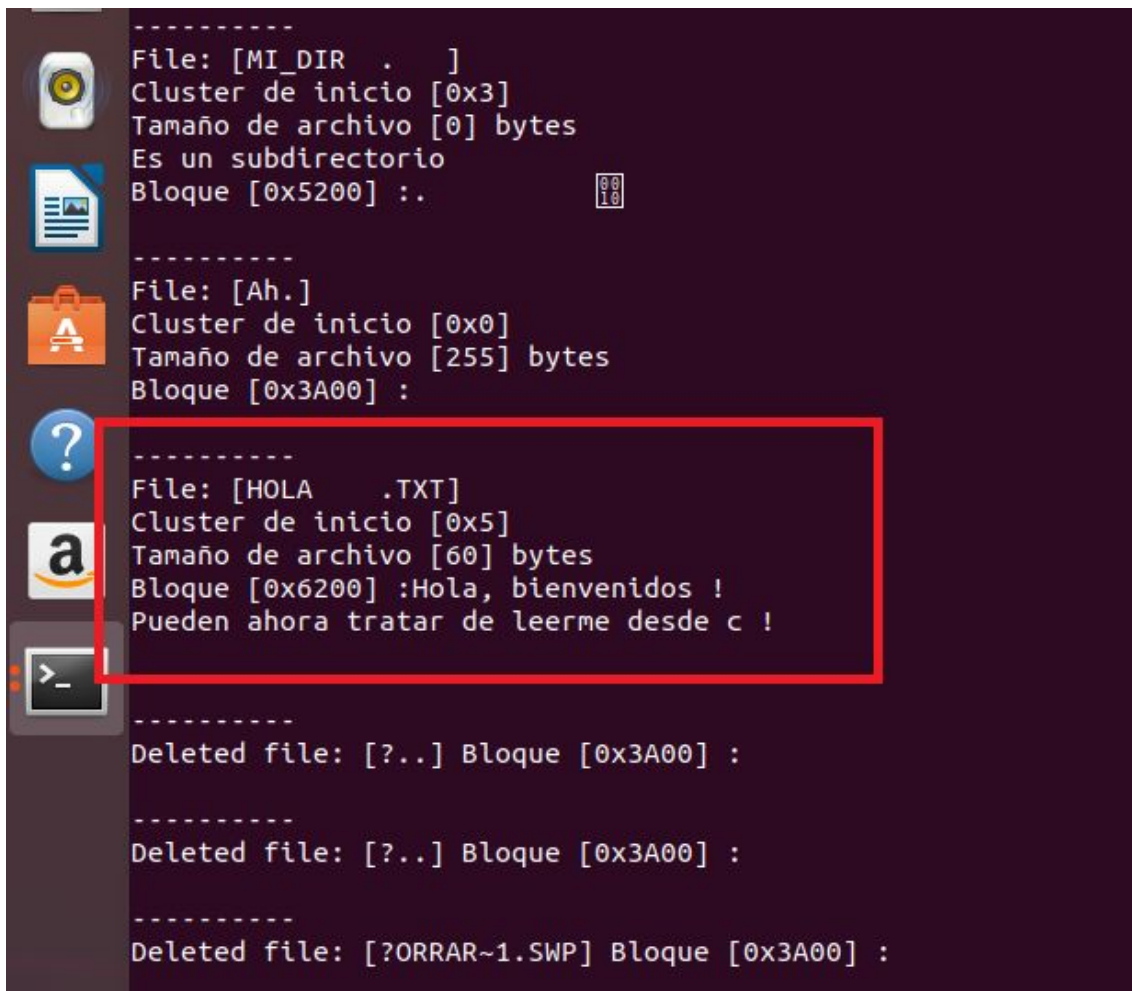
#### 4. Leyendo archivos.

a) Muestre, mediante el hex editor y mediante código C lo que hay en el archivo no borrado.

Sabemos que el archivo "HOLA.TXT" apunta al cluster de inicio 5. Teniendo esa información, junto con el tamaño del cluster y la posición del disco a partir de donde comienzan los datos,

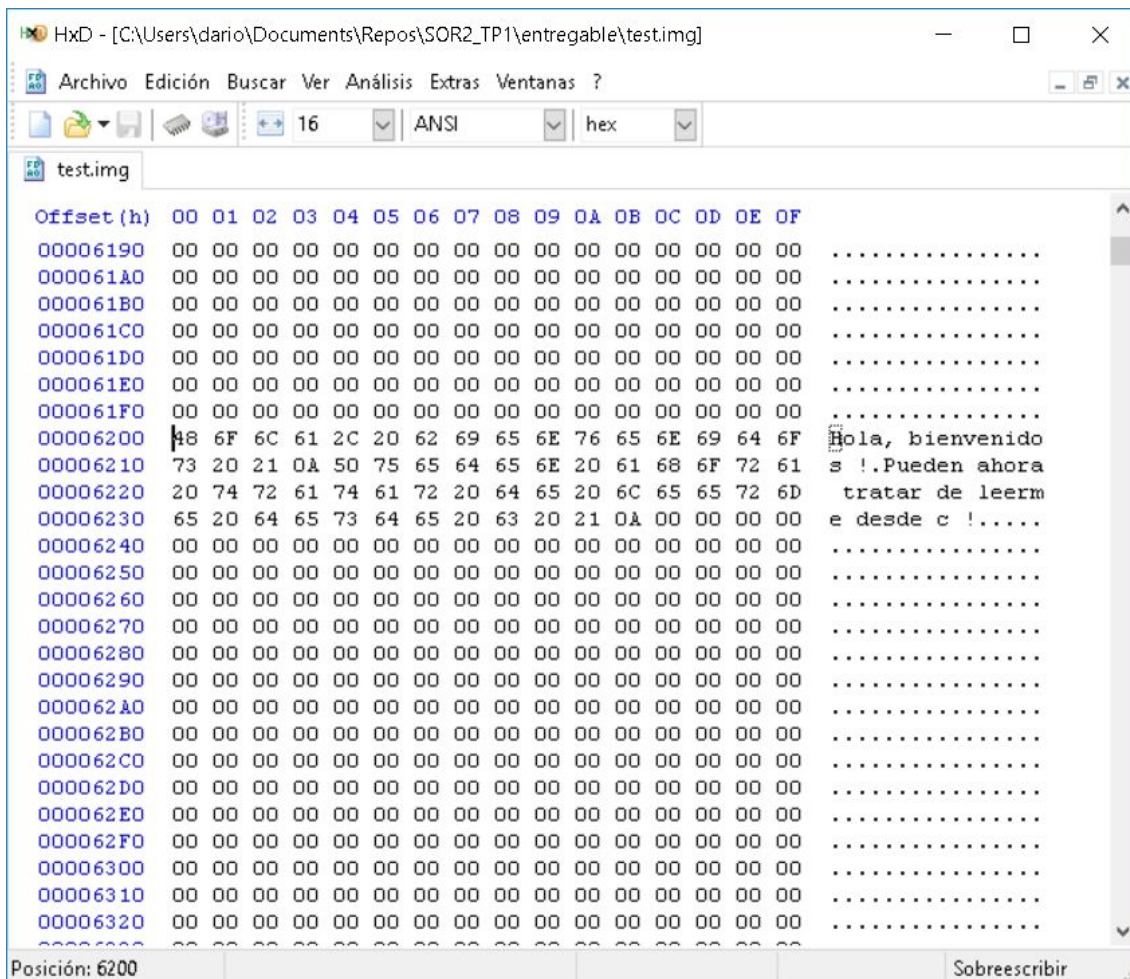


podemos ir a buscar el contenido del mismo



A terminal window with a dark purple background and a vertical sidebar on the left containing icons for a hard drive, a document, a shopping bag, a question mark, an Amazon logo, and a terminal icon. The terminal displays several file entries in a monospaced font. A red rectangular box highlights the entry for 'HOLA.TXT'. The text in the terminal is as follows:

```
-----  
File: [MI_DIR . ]  
Cluster de inicio [0x3]  
Tamaño de archivo [0] bytes  
Es un subdirectorio  
Bloque [0x5200] :.  
  
-----  
File: [Ah.]  
Cluster de inicio [0x0]  
Tamaño de archivo [255] bytes  
Bloque [0x3A00] :  
  
-----  
File: [HOLA .TXT]  
Cluster de inicio [0x5]  
Tamaño de archivo [60] bytes  
Bloque [0x6200] :Hola, bienvenidos !  
Pueden ahora tratar de leerme desde c !  
  
-----  
Deleted file: [?..] Bloque [0x3A00] :  
  
-----  
Deleted file: [?..] Bloque [0x3A00] :  
  
-----  
Deleted file: [?ORRAR~1.SWP] Bloque [0x3A00] :
```



**b) Modifique el filesystem para que el archivo borrado sea recuperado. (Puede hacerlo mediante el hex editor)**

Si se cambia el byte E5 a 05, el archivo puede ser recuperado.

[illegible]

```
root@dario-VirtualBox:/home/dario/Repos/SOR2_TP1/entregable# mount test.img /mnt/img -t msdos -o loop,fat=12
root@dario-VirtualBox:/home/dario/Repos/SOR2_TP1/entregable# ls /mnt/img/
hola.txt  mi_dir  '$\345'orrr~1.swp
root@dario-VirtualBox:/home/dario/Repos/SOR2_TP1/entregable#
```