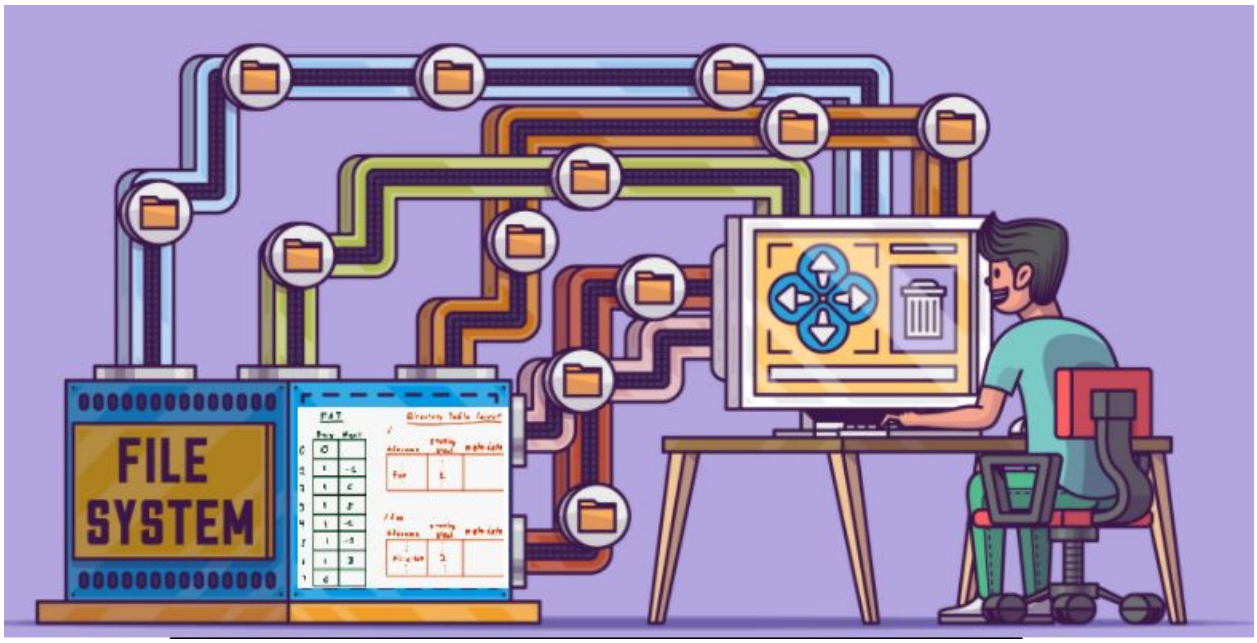
Universidad Nacional de General Sarmiento

Sistemas Operativos y Redes 2

Trabajo práctico 1: FAT



Profesores:

Alexander, Agustín
Gutierrez, Pedro

Alumnos:

Costilla, Germán Alfredo
Dougan, Vanesa Solange

Contenido

Ejercicio 1:

1. Al montarlo. ¿ Para qué se ha puesto umask=000 ?

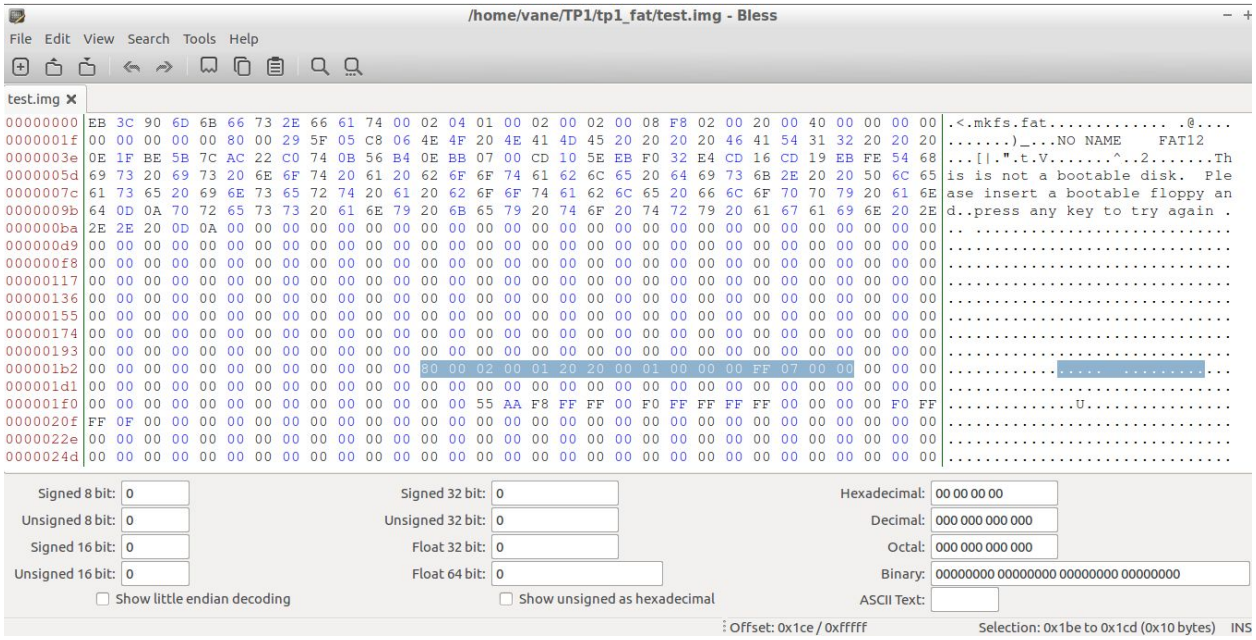
Al crear un archivo en linux su permiso por defecto es 0777. El comando umask sirve para controlar cuál será el valor por defecto al crear un archivo. Este consiste en aplicarle una mascara a 0777 y a partir de su resultado establecer los permisos. Por lo tanto, el valor resultante de los permisos del archivo termina siendo: valor original - mascara.
Se ha puesto umask 000 para poder tener permisos de lectura, escritura y ejecución, ya que 000 no quita nada.
Los permisos de las máscaras no son como los códigos de permisos octales pasados al comando chmod. Funciona como el código octal de chmod pero restando 7 y usando el valor absoluto. Por ejemplo, si se quiere configurar permisos 0777 se necesitará configurar 0000 en unmask (umask=0000), si se quiere configurar 0755 , se deberá configurar 0022

- El primer caracter representa que es un permiso octal
- El segundo es el owner
- El tercero es el grupo
- El cuarto es para otro

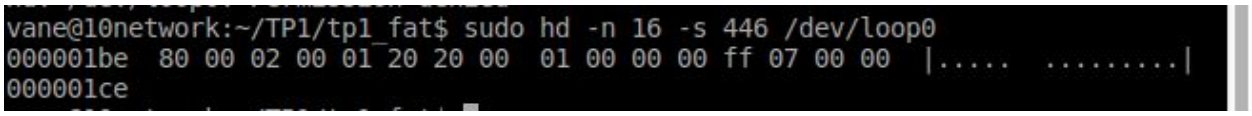
Ejercicio 2: Cargando el MBR.

a) Mostrando 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.

Hay 4 particiones, la primera va de 446-461, la segunda 462-477, la tercera 478-493 y la cuarta 494-509. Cada una de 16 bytes.
Sólo se encuentra activa la primera partición (446-461) mientras que las demás están vacías con 00.



Por medio de la consola se puede encontrar:



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

Por medio del código de C del archivo read_boot.C se puede ver información como la cantidad de FATs, el tamaño de los sectores de FAT, el tipo de File System.

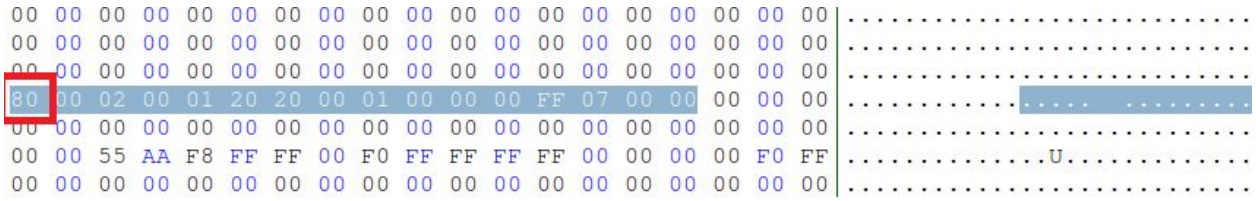
```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ ./read_boot
Partition type: 1
Encontrado FAT12 0
  Jump code: EB:3C:90
  OEM code: [mkfs.fat]
  sector_size: 512
  Fat Size Sectors: 2
  Quantity of FATs: 2
  Reserved Sectors: 1
  Root Directory Entries: 512
  volume_id: 0x06C8055F
  Volume label: [NO NAME    ]
  Filesystem type: [FAT12   ]
  Boot sector signature: 0xAA55
```

El código que se utiliza para mostrar en C las particiones del MBR, se encuentran en el archivo C llamado read_mbr.c

```
gercos@gercos:~/Documentos/SOR2/tp1_Fat$ ./read_mbr
Partition entry 0: First byte 80
  Comienzo de partición en CHS: 00:02:00
  Partition type 0x01
  Fin de partición en CHS: 00:20:20
  Dirección LBA relativa 0x00000001, de tamaño en sectores 2047
Partition entry 1: First byte 00
  Comienzo de partición en CHS: 00:00:00
  Partition type 0x00
  Fin de partición en CHS: 00:00:00
  Dirección LBA relativa 0x00000000, de tamaño en sectores 0
Partition entry 2: First byte 00
  Comienzo de partición en CHS: 00:00:00
  Partition type 0x00
  Fin de partición en CHS: 00:00:00
  Dirección LBA relativa 0x00000000, de tamaño en sectores 0
Partition entry 3: First byte 00
  Comienzo de partición en CHS: 00:00:00
  Partition type 0x00
  Fin de partición en CHS: 00:00:00
  Dirección LBA relativa 0x00000000, de tamaño en sectores 0
```

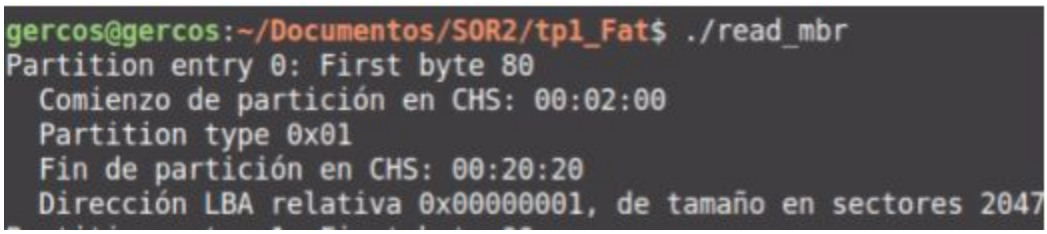
c) Muestre en el Hex Editor si la primer participación es booteable o no. ¿Lo es?

La primera partición es booteable ya que contiene al inicio el número 80. El valor 80 indica que una partición está activa, es dónde está configurado el flag del boot. Si la partición no está activa tiene el número 00.



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

El código que se utiliza para mostrar en C las particiones del MBR, se encuentran en el archivo C llamado read_mbr.c

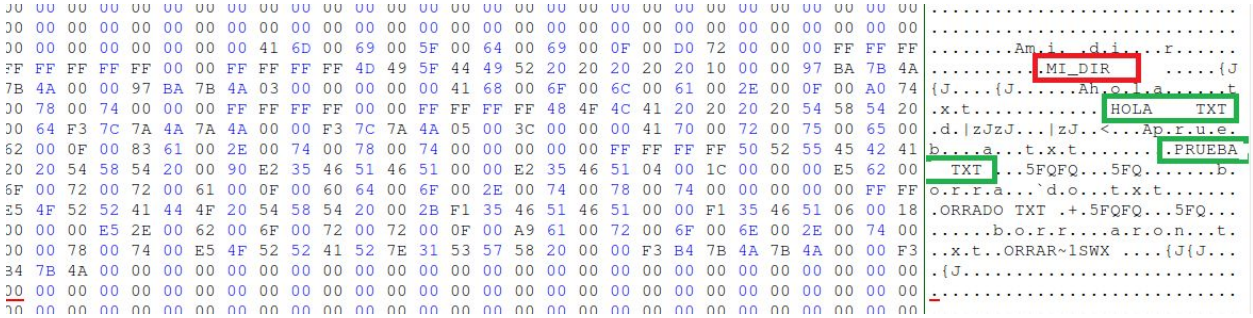


El tipo de partición 0x01 significa FAT 12.

Ejercicio 3: Cargando la tabla de archivos.

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

A continuación, se muestra utilizando Bless los archivos que tiene el fileSystem, por un lado MI_DIR (en rojo) que corresponde a un directorio y por otro lado, hola.txt y prueba.txt (en verde) que corresponde a los archivos.



Utilizando el código C, a continuación se muestran los archivos/directorios encontrados. (los mismos que se encontraron en Bless). Además se muestra un listado de archivos que fueron borrados.

```
gercos@gercos:~/Documentos/SOR2/tp1_Fat$ ./read_root
Encontrada particion FAT12 0
En  0x200, sector size 512, FAT size 2 sectors, 2 FATs

Root dir entries 512
Directorio: [MI_DIR  .   ]
Archivo: [HOLA    .TXT]
Archivo: [PRUEBA  .TXT]
Archivo borrado: [?b.]
Archivo borrado: [?ORRADO .TXT]
Archivo borrado: [?..]
Archivo borrado: [?ORRAR~1.SWX]

Leido Root directory, ahora en 0x4A00
```

El código para obtener este listado, se encuentra en el archivo C llamado read_root.C. Se indica si la entrada es un archivo o directorio.

b) Montando el filesystem (mediante mount) cree un archivo en la carpeta root / y luego bórrelo. Búsquelo por bless y muéstrelo en con el código generado previamente.

Se procede a crear el archivo “creado.txt”.

```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ echo 'hola mundo estoy creando un archivo' > /mnt/creado.txt
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ cat /mnt/creado.txt
hola mundo estoy creando un archivo
```

En Bless se puede ver que aparece el archivo:

```
64 00 69 00 0F 00 D0 72 00 00 00 FF FF FF FF FF FF FF FF 00 00 FF FF FF FF |....Am.i._.d.i....r.............|
10 00 00 97 BA 7B 4A 7B 4A 00 00 97 BA 7B 4A 03 00 00 00 00 00 41 68 00 6F |MI_DIR     .....{J{J....{J......Ah.|
00 78 00 74 00 00 00 FF FF FF FF 00 00 FF FF FF FF 48 4F 4C 41 20 20 20 20 |.l.a......t.x.t.............HOLA |
4A 00 00 F3 7C 7A 4A 05 00 3C 00 00 00 41 70 00 72 00 75 00 65 00 62 00 0F |TXT .d.|zJzJ...|zJ..<...Ap.r.u.e.b.|
00 00 00 00 00 FF FF FF FF 50 52 55 45 42 41 20 20 54 58 54 20 00 90 E2 35 |..a...t.x.t.........PRUEBA  TXT ...|
00 1C 00 00 00 41 63 00 72 00 65 00 61 00 64 00 0F 00 67 6F 00 2E 00 74 00 |FQFQ...5FQ......Ac.r.e.a.d...go...t|
FF 43 52 45 41 44 4F 20 20 54 58 54 20 00 64 F5 A2 59 51 59 51 00 00 F5 A2 |x.t.........|CREADO  TXT |.d..YQYQ...|
62 00 6F 00 72 00 72 00 0F 00 A9 61 00 72 00 6F 00 6E 00 2E 00 74 00 00 00 |YQ..$......b.o.r.r....a.r.o.n...t..|
31 53 57 58 20 00 00 F3 B4 7B 4A 7B 4A 00 00 F3 B4 7B 4A 00 00 00 00 00 00 |x.t..ORRAR~1SWX ....{J{J....{J.....|
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |................................|
```

Por medio del código C, también se puede ver que se creó el archivo. Ejecutando el código read_root.C.

```
Root dir_entries 512
Directorio: [MI_DIR  .   ]
Archivo: [HOLA    .TXT]
Archivo: [PRUEBA  .TXT]
Archivo: [CREADO  .TXT]
```

c) Muestre mediante bless el archivo que ha sido borrado. Explique cómo lo ha visto. Genere código C para mostrarlos.

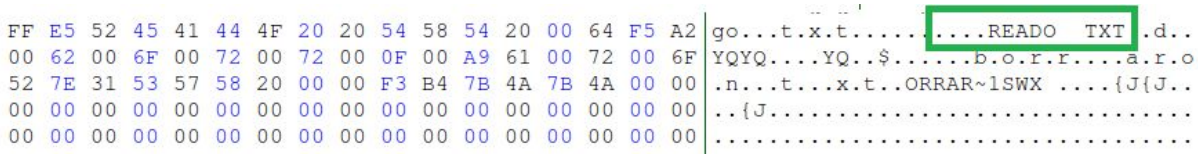
Se procede a borrar el archivo:

```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ rm /mnt/creado.txt
```

Por medio del código C, se puede ver que esta borrado:

```
Root dir entries 512
Directorio: [MI_DIR  .   ]
Archivo: [HOLA    .TXT]
Archivo: [PRUEBA  .TXT]
Archivo que comienza con 0xE5: [?c.]
Archivo que comienza con 0xE5: [?READO  .TXT]
Archivo que comienza con 0xE5: [?..]
Archivo que comienza con 0xE5: [?ORRAR~1.SWX]
```

En Bless también se puede ver que el archivo fue borrado:



d) ¿Qué puede decir acerca del recupero de archivos ?

Cuando un archivo es borrado dentro de FAT, su información sigue existiendo en disco, lo que pasa realmente que se renombra, es decir, se marca para ser ignorado, reemplazando el primer carácter de su nombre por 0xE5. En cuanto a la meta-data (fecha y hora de creación, permisos, tamaño y lo más importante la dirección del inicio del cluster del archivo) del archivo eliminado sigue existiendo, lo cual hace posible poder recuperarlo ya que tenemos la información del cluster donde se encuentra su contenido. Para poder hacer el recupero de archivo:

- Leemos las entradas del root directory que inician con 0xE5
- Renombramos el primer carácter de cada una de estas entradas por uno valido, ya que no sabemos que tenía originalmente.

Ejercicio 4. Leyendo archivos.

a) Montando el filesystem (mediante mount) cree un archivo llamado lapapa.txt y póngale algún texto como contenido. Hágalo en la carpeta root /. . Búsquelo por bless y muéstrelo con el código generado previamente.

Se procede a crear el archivo lapapa.txt.

```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ echo 'hola soy el contenido de la papa' > /mnt/lapapa.txt
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ cat /mnt/lapapa.txt
hola soy el contenido de la papa
```

Por medio del código C, se puede ver que el archivo lapapa.txt fue creado.

```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ ./read_root
Encontrada particion FAT12 0
En  0x200, sector size 512, FAT size 2 sectors, 2 FATs

Root dir_entries 512
Directorio: [MI_DIR   .    ]
Archivo: [HOLA     .TXT]
Archivo: [PRUEBA   .TXT]
Archivo: [LAPAPA   .TXT]
Archivo que comienza con 0xE5: [??..]
Archivo que comienza con 0xE5: [??ORRAR~1.SWX]
```

También se puede verificar en Bless:

```
FF FF FF FF FF FF FF FF 00 00 FF FF FF FF 4D 49 5F 44 49 52 20 20 Am.i._.d.i....r.................MI_DIR
03 00 00 00 00 00 41 68 00 6F 00 6C 00 61 00 2E 00 0F 00 A0 74 00    .....{J{J....{J......Ah.o.l.a......t.
4C 41 20 20 20 20 54 58 54 20 00 64 F3 7C 7A 4A 7A 4A 00 00 F3 7C x.t.............HOLA    TXT .d.|zJzJ...|
00 0F 00 83 61 00 2E 00 74 00 78 00 74 00 00 00 00 00 FF FF FF FF zJ..<...Ap.r.u.e.b....a...t.x.t.........
46 51 00 00 E2 35 46 51 04 00 1C 00 00 00 41 6C 00 61 00 70 00 61 PRUEBA  TXT ...5FQFQ...5FQ......Al.a.p.a
00 00 FF FF FF FF 4C 41 50 41 50 41 20 20 54 58 54 20 00 00 97 A6 .p....a...t.x.t........LAPAPA  TXT  ...
00 62 00 6F 00 72 00 72 00 0F 00 A9 61 00 72 00 6F 00 6E 00 2E 00 YQYQ....YQ..!......b.o.r.r....a.r.o.n...
58 20 00 00 F3 B4 7B 4A 7B 4A 00 00 F3 B4 7B 4A 00 00 00 00 00 00 t...x.t..ORRAR~1SWX ....{J{J....{J......
```

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

Por medio de Bless, se puede ver el contenido que tiene el archivo lapapa.txt:

```
0 00 00 00
```

código se recuperaron archivos, entre ellos el “lapapa.txt”. También, se puede observar que se le coloca en el primer carácter del nombre del archivo un `.

```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ ./recovery
Encontrada particion FAT12 0
En  0x200, sector size 512, FAT size 2 sectors, 2 FATs

Root dir entries 512
Directorio: [MI_DIR  .   ]
Archivo: [HOLA    .TXT]
Archivo: [PRUEBA  .TXT]
Archivo borrado: [?l.]
Entre al recoverSize del fat12entry 32position 2752
Archivo recuperado `l
Archivo borrado: [?APAPA  .TXT]
Entre al recoverSize del fat12entry 32position 2784
Archivo recuperado `APAPA  TXT
Archivo borrado: [?..]
Entre al recoverSize del fat12entry 32position 2816
Archivo recuperado `.
Archivo borrado: [?ORRAR~1.SWX]
Entre al recoverSize del fat12entry 32position 2848
Archivo recuperado `ORRAR~1SWX
```

Por último se verifica con read_root que los archivos fueron recuperados.

```
vane@10network:~/TP1/TP1_FAT_COSTILLA_DOUGAN$ ./read_root
Encontrada particion FAT12 0
En  0x200, sector size 512, FAT size 2 sectors, 2 FATs

Root dir entries 512
Directorio: [MI_DIR  .   ]
Archivo: [HOLA    .TXT]
Archivo: [PRUEBA  .TXT]
Archivo: [`APAPA  .TXT]
Archivo: [`ORRAR~1.SWX]
```