

Sistema de Archivos – Laboratorio.

Material de Consulta

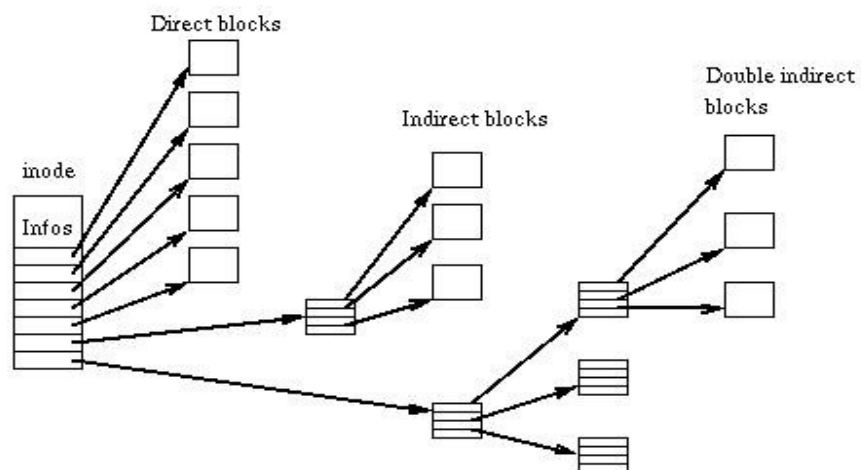
- Sistemas Operativos Modernos. Andrew S. Tanenbaum 1992 Prentice Hall.
- Guía práctica de FS de la cátedra (Repositorio, página de la cátedra)
- http://www.barnech.com/uai-sistemas/arg_so/Linux/sistema_de_archivos_ext2.htm
- <http://www.win.tue.nl/~aeb/linux/fs/ext2/ext2.html#AEN18>
- /usr/include/linux/ext2_fs.h
- /usr/include/linux/ext2_fs_sb.h
- http://lde.sourceforge.net/lde_use.html
- Maps for Ext2 FS (Repositorio, página de la cátedra)

El sistema de archivos EXT2 (Second Extended File System)

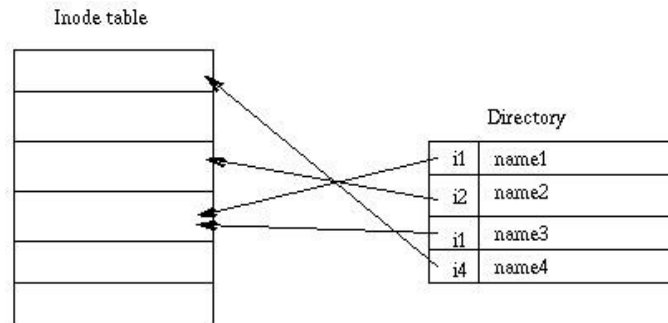
Algunos conceptos o premisas básicas:

➔ **ARCHIVOS:** son representados por una estructura llamada Inodo:

- Cada archivo se describe por un inodo solamente.
- Cada inodo tiene un número único que lo identifica. Se almacenan en la Tabla de Inodos.
- Un inodo contiene la descripción del archivo: tipo, derechos de acceso, propietarios, etc y punteros a los bloques de datos. La siguiente figura es una representación del campo de direcciones de los bloques de datos dentro de un inodo:



- **DIRECTORIOS:** son "archivos" que contienen una lista de entradas (que apuntan a otros directorios o archivos). Estructura de un directorio:



- **DISPOSITIVOS DE E/S:** pueden ser accedidos a través de ficheros especiales.

Estructura física de un FS EXT2:

La "Organización del disco" se basa en **Bloques** y **Grupos de Bloques**.

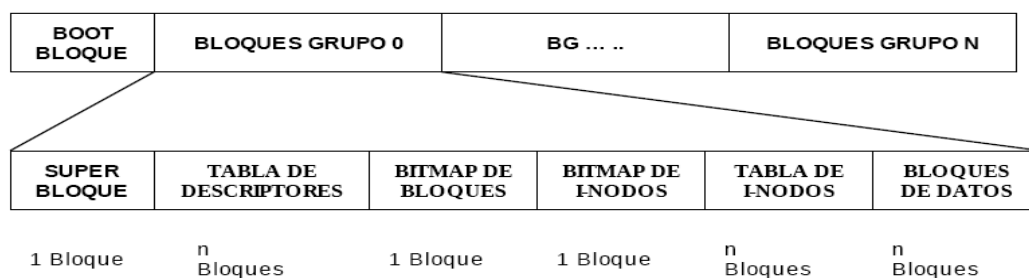
→ GRUPO DE BLOQUES:

- El FS se compone de grupos de bloques (estos grupos no se encuentran atados a la disposición física de los bloques en el disco).
- Ext2 divide las particiones lógicas que ocupa, en Grupo de Bloques (BG): conjunto de bloques secuenciales.
(Mantiene la información relacionada físicamente cercana facilitando las tareas de gestión).
- Qué contiene cada Grupo de Bloque:
 - Una copia del Superbloque
 - Tabla con los descriptores del grupo
 - Bitmap de bloques
 - Bitmap de inodos
 - Parte de la tabla de inodos
 - Bloques de datos

→ BLOQUES lógicos:

- Es la unidad más pequeña de almacenamiento que puede ser asignada por el FS.
- Tamaño de un bloque: todos los bloques son de la misma longitud - se miden en bytes (esa longitud se decide al momento de crear un FS EXT2 en particular, utilizando **mke2fs** –en el man puede encontrar más información. En un disquete por ejemplo, un bloque puede ser de 1 KB mientras que en una partición de 10 GB habitualmente sería de tamaño 8 KB ó 4 KB.
- Qué almacenan: todos los datos que contienen los archivos se guardan en "bloques" de datos (el tamaño de cada archivo se redondea hasta un número entero de bloques). Algunos bloques se utilizan para guardar la información de la estructura del FS.

Estructura de **Bloques** y **Grupos de Bloques**:



→ DIRECTORIO: se muestra a continuación la estructura de un directorio:

inode number	entry length	name length	filename
-----------------	-----------------	----------------	----------

Layout del FS.

→ En un disquete:

Offset	# of blocks	description
0	1	boot record
----- block group 0 -----		
(1024bytes)	1	superblock
2	1	group descriptors
3	1	block bitmap
4	1	inode bitmap
5	23	inode table
28	1412	data blocks

→ En una partición de disco de 20 MB:

Offset	# of blocks	description
0	1	boot record
----- block group 0 -----		
(1024 bytes)	1	Superblock
2	1	group descriptors
3	1	block bitmap
4	1	inode bitmap
5	214	inode table
219	7974	data blocks
----- block group 1 -----		
8193	1	Superblock
8194	1	group descriptors
8195	1	block bitmap
8196	1	inode bitmap
8197	214	inode table
8408	7974	data blocks
----- block group 2 -----		
16385	1	Superblock
16386	1	group descriptors
16387	1	block bitmap
16388	1	inode bitmap
16389	214	inode table
16601	3877	data blocks

Datos y Estructuras de datos:

La definición de las estructuras y algunas constantes destacadas se describen a continuación, pueden ser consultadas en forma completa y accedidas a través de:

`/usr/include/linux/ext2_fs.h`

(se resaltan en azul las estructuras o datos identificados en la descripción del layout presentado anteriormente)

→ `struct ext2_super_block`

. Contiene información del "Superbloque"

. Superbloque: es el bloque con información más relevante. Tiene la descripción del tamaño y forma del FS. Normalmente sólo se lee el SuperBloque del Grupo de Bloque 0, cuando se monta el FS (igualmente cada Grupo de bloque contiene una copia del Superbloque por si el FS se corrompe. Se encuentra en el offset fijo 1024 del disco y ocupa 1024 bytes).

.Algunos datos del Superbloque:

- Tamaño total del sistema de archivos, en bloques o nodos-i.
- Número de bloques libres del sistema.
- Número de bloques reservados a nodos-l.
- Número de nodos-l libres.
- Dirección del primer bloque de datos.
- Tamaño de un bloque de datos.
- Tamaño de un bloque parcial de datos.
- Hora de la última modificación sistema archivos.
- Hora integración (montaje) del sistema.
- Número de versión del sistema.
- Hora de la última verificación del sistema.

→ `struct ext2_group_desc`

. La tabla de descriptores de grupo, se define a través de esta estructura.

. Descriptores de grupo: se colocan todos juntos para formar la tabla, que es almacenada a continuación del superbloque. En cada BG(Block Group) hay una

copia de esta tabla. Sólo se utiliza la del BG 0, las otras se leerán en caso de que el FS esté dañado.

Los descriptores tienen información para gestionar bloques e inodos en cada grupo.

.Algunos datos de esta estructura:

- [bg_block_bitmap](#)
Dirección del bloque de bitmap de bloques.
(Se referencia para reservar y liberar bloques)
- [bg_inode_bitmap](#)
Dirección del bloque de bitmap de inodos.
(Se referencia para reservar y liberar inodos)
- [bg_inode_table](#)
Dirección de la tabla de inodos.
Cada inodo se representa por una estructura:
struct ext2_inode
- Número de bloques libres.
- Número de inodos libres.

➔ [EXT2_ROOT_INO](#)

El Directorio Raíz es siempre la segunda entrada de la Tabla de Inodos. A partir de ahí cualquier archivo o subdirectorio puede ser localizado.

El **bloque de datos** almacena el contenido de archivos, incluyendo: lista de directorios, atributos, links simbólicos, etc.

Como ya se mencionó los directorios se almacenan como archivos y pueden ser identificados a través del contenido `ext2_inode_i_mode`.

La estructura que tiene la información de las entradas de directorio es:

struct ext2_dir_entry

Analizando el "FS – ext2" de un disquete:

1) Crear un FS tipo ext2 sobre un disquete:

```
#mkfs.ext2 /dev/fd0
```

La salida de la ejecución del comando anterior, mostrará la siguiente información por salida estándar:

```
Etiqueta del sistema de ficheros=
Tipo de SO: Linux
Tamaño del bloque=1024 (bitácora=0)
Tamaño del fragmento=1024 (bitácora=0)
184 nodos i, 1440 bloques
72 bloques (5.00%) reservados para el súper usuario
Primer bloque de datos=1
Maximum filesystem blocks=1572864
1 bloque de grupo
8192 bloques por grupo, 8192 fragmentos por grupo
184 nodos i por grupo

Mientras se escribían las tablas de nodos i: 0/1__terminado
Escribiendo superbloques y la información contable del sistema de
ficheros: hecho

Este sistema de ficheros se revisará; automáticamente cada 26 meses o
180 días, lo que suceda primero. Utilice tune2fs -c o -i para cambiarlo.
```

Por defecto se crearán bloques de 1024 bytes (total de 1440 bloques).

El tamaño de bloque puede ser especificado al momento de crear el FS en el dispositivo, por ejemplo bloques de 2K:

```
# mke2fs -b 2048 /dev/fd0
```

```
mke2fs 1.41.12 (17-May-2010)
warning: Unable to get device geometry for /dev/fd0
Filesystem label=
OS type: Linux
Block size=2048 (log=1)
Fragment size=2048 (log=1)
Stride=0 blocks, Stripe width=0 blocks
192 inodes, 720 blocks
36 blocks (5.00%) reserved for the super user
First data block=0
1 block group
16384 blocks per group, 16384 fragments per group
192 inodes per group

Writing inode tables: done
Writing superblocks and filesystem accounting information: done
```

```
This filesystem will be automatically checked every 32 mounts or  
180 days, whichever comes first. Use tune2fs -c or -i to override.
```

Puede verse que ahora se tienen 720 bloques.

Verificando algunos números:

- Para bloques de tamaño: 1024 bytes se obtienen 1440 bloques
→ $1024 \text{ (bytes)} * 1440 \text{ (nro. bloques)} = 1440 \text{ KB}$ (tamaño físico del dispositivo)
- Para bloques de tamaño: 2048 bytes se obtienen 720 bloques
→ $2048 \text{ (bytes)} * 720 \text{ (nro. bloques)} = 1474560 \text{ bytes} = 1440 \text{ KB}$ (tamaño físico del dispositivo)

2) Utilizando el LDE (Linux Disk Editor) se puede visualizar y editar el FS en un dispositivo:

(puede obtener información de cómo utilizar **lde** en el link sugerido en el material de consulta o consultando el manual: **man lde**)

#lde /dev/fd0

- o lo primero que informará el editor es si ha detectado un FS y de qué tipo.
- o Presionando cualquier tecla se va a la próxima pantalla:

Esta pantalla contiene parte de la información del "Superbloque"


```
File Edit View Terminal Help
lde v2.6.0 : ext2 : /dev/fd0
node:      2 (0x00000002) Block:      0 (0x00000000) 0123456789!@#$%^

Inodes:      184 (0x000000B8)
Blocks:      1440 (0x000005A0)
Firstdatazone: 1 (N=1)
Zonesize:    1024 (0x0400)
Maximum size: 16843020 (0x0101010C)

* Directory entries are 255 characters.
* Inode map occupies 1 blocks.
* Zone map occupies 1 blocks.
* Inode table occupies 23 blocks.

F)lags, I)node, B)locks, R)ecover File
```

3) Comparando información del punto 1 y 2.

Se puede observar que, parte de la información que arroja el comando para crear el FS sobre el dispositivo, está contenida en el Superbloque (esto puede leerse con el lde en la pantalla presentada anteriormente).

Algunos datos.

- Cantidad de Inodos: 184
- Total de bloques: 1440
(si se suma la columna #of blocks en el layout del disquete verá que es 1440)
- Tamaño de bloque: 1024 bytes
(en el lde se identifica como Zonesize)
- Tabla de Inodos ocupa: 23 bloques

4) Calcular el Tamaño máximo de FS para este dispositivo:

TAMAÑO MÁXIMO DE ARCHIVO = Tamaño de bloque * Nros_Punteros

- 1) Tamaño de bloque = 1 Kb (ver punto anterior)

2) Nros_Punteros = mínimo { punteros s/direccionamiento , punteros s/estructura i-nodo }

2.1) Cantidad de punteros según el direccionamiento: 2^n

$n = 32 \text{ bits} = 2^{32} = \mathbf{4.294.967.296 \text{ punteros}}$

n es definido por el Sistema de Archivos

¿De dónde obtenemos esta información?

n, por definirlo de algún modo, es el tamaño con que se van a establecer las direcciones. Las direcciones son los punteros.

- Los PUNTEROS A BLOQUES están definidos en la estructura de inodo:

```
struct ext2_inode{
    ..
    ..
    __le32 i_block[EXT2_N_BLOCKS]; /*Pointers to blocks*/
    ..
}
```

De ese campo podemos leer dos cosas:

- 1) `__le32` (Little-endian 32 bits) → $n=32 \text{ bits}$
- 2) `EXT2_N_BLOCKS`: nos dará la cantidad de punteros que hay a bloques de datos. El total es 15 punteros.

¿Cómo obtenemos esta información?

Está definida en `ext2_fs`:

<code>EXT2_NDIR_BLOCKS</code>	→ 12	/* número de bloques directos */
<code>EXT2_IND_BLOCK</code>	→ <code>EXT2_NDIR_BLOCKS</code>	/* bloque simple-indirecto*/
<code>EXT2_DIND_BLOCK</code>	→ <code>(EXT2_IND_BLOCK + 1)</code>	/* bloque doble-indirecto*/
<code>EXT2_TIND_BLOCK</code>	→ <code>(EXT2_DIND_BLOCK + 1)</code>	/* bloque triple-indirecto*/
<code>EXT2_N_BLOCKS</code>	→ <code>(EXT2_TIND_BLOCK + 1)</code>	/* número total de bloques*/

El modo de interpretar la información del array `i_block[]` es la siguiente:

- `i_block[0..11]`** puntero directo a los primeros 12 bloques de datos del archivo.
- `i_block[12]`** puntero a un bloque simple-indirecto
- `i_block[13]`** puntero a un bloque doble-indirecto
- `i_block[14]`** puntero a un bloque triple-indirecto

(cada elemento del array es un puntero y el array tiene 15 elementos)

2.2) Cantidad de punteros según la estructura del i-nodo:

Tamaño de bloque = 1 KB

Tamaño de dirección de disco = 32 bits = 4 bytes (demostrado en el punto anterior)

Punteros almacenados por bloque = Tamaño de bloque / Tam_dirección

Punteros almacenados por bloque = (1024 bytes) / 4 bytes = 256 punteros a bloque

Nodo-i posee:

- 12 punteros a bloques directos
- 1 bloque simple-indirecto = 256 (cada bloque puede almacenar 256 punteros)
- 1 bloque doble-indirecto = $(256)^2$ (el bloque doblemente indirecto puede almacenar 256 punteros a bloques que contienen 256 punteros)
- 1 bloque triple-indirecto = $(256)^3$

Por lo que:

Cant. de punteros s/la estructura del i-nodo = $12 + 256 + (256)^2 + (256)^3 = 16.843.020$ punteros

➔ Obtengo la cantidad de punteros de acuerdo a lo determinado en el punto 2)

Nros_Punteros = mínimo { punteros s/direccionamiento , punteros s/estructura i-nodo }

Nros_Punteros = mínimo { 4.294.967.296 , 16.843.020 } = **16.843.020 punteros**



El valor obtenido podemos verificarlo en la información del superbloque brindada por el lde (Maximun Size)

Reemplazando en la fórmula inicial:

$$TMFS = 1 \text{ [KB]} * 16.843.020 \text{ [punteros]} = 16.843.020 \text{ [KB]}$$

PERO, como específicamente queremos conocer el TMA para una disquete de 1440 KB

$$TMFS = \min \{ \text{Tamaño dispositivo, Tamaño según la estructura del FS} \}$$

$$TMFS = \min \{ 1440 \text{ [KB]}, 16.843.020 \text{ [KB]} \}$$

$$TMFS = 1440 \text{ [KB]}$$

5) Recorriendo el mapa del FS ext2 (utilizando LDE: Linux Disck Editor)

Algunas opciones básicas:

- Tecla **S**: Superbloque
- Tecla **I**: inodos (pg_down – pg_up: para avanzar o retroceder por inodos)
(shift + #: permite ir al número de inodo determinado)
- Tecla **B**: bloques (pg_down – pg_up: para avanzar o retroceder por bloques)
(shift + #: permite ir al número de bloque determinado)
- Tecla **Q**: salir

S → Superbloque:

Al acceder a **lde** luego de identificar e indicar cuál es el FS del dispositivo, presionando cualquier tecla encontramos parte de la información contenida en el Superbloque (estando en algún otro lugar, presionando **S** volvemos a esta pantalla).

I → Recorrido por Inodos:

En la barra superior, el primer dato a la izquierda es:

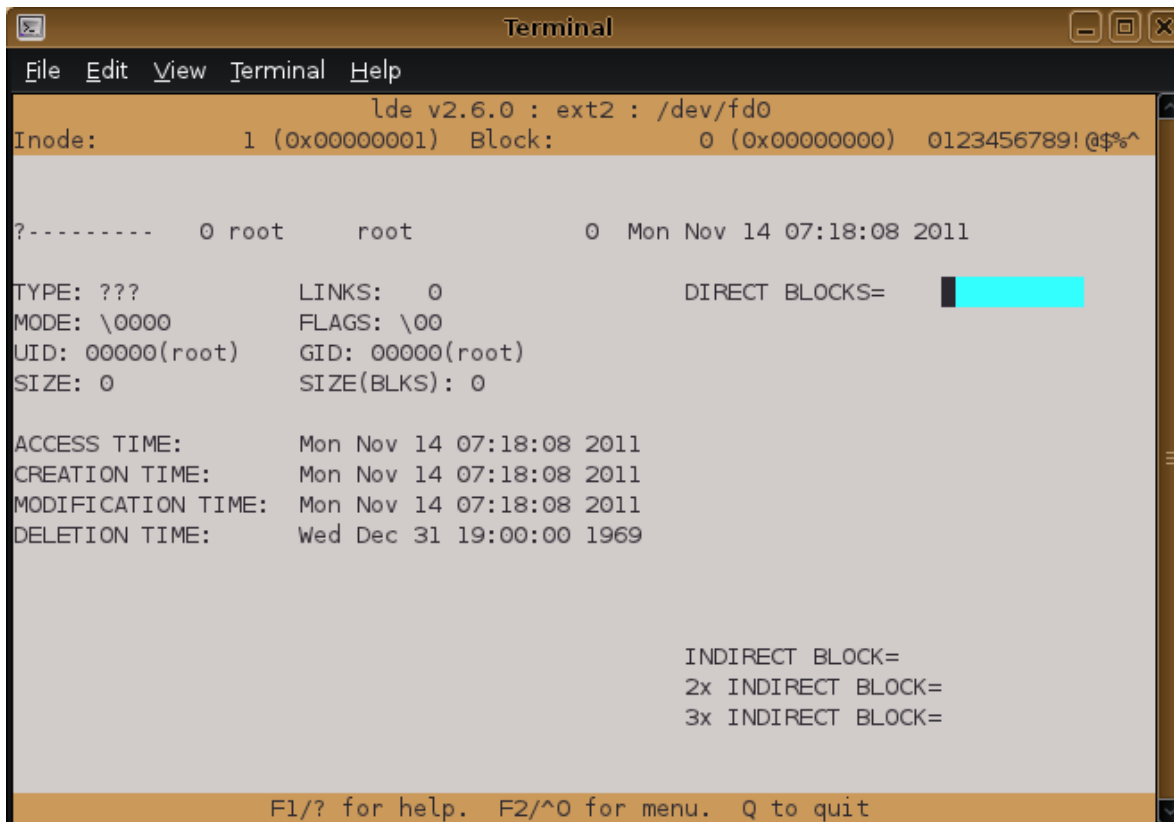
Inode:	número de inode
--------	-----------------

Inodo: 1

La primera entrada de la tabla de inodos está reservada, si vemos la definición en ext2_fs.h:

```
EXT2_BAD_INO → 1 /* Bad blocks inode */
```

Este inodo no tiene links, ni punteros a bloques de datos.



```
Terminal
File Edit View Terminal Help
ls -ld v2.6.0 : ext2 : /dev/fd0
Inode:      1 (0x00000001)  Block:      0 (0x00000000)  0123456789!@#$%^
?-----  0 root      root      0  Mon Nov 14 07:18:08 2011
TYPE: ???      LINKS:    0      DIRECT BLOCKS=
MODE: \0000    FLAGS: \00
UID: 00000(root)  GID: 00000(root)
SIZE: 0      SIZE(BLKS): 0
ACCESS TIME:    Mon Nov 14 07:18:08 2011
CREATION TIME:  Mon Nov 14 07:18:08 2011
MODIFICATION TIME: Mon Nov 14 07:18:08 2011
DELETION TIME:  Wed Dec 31 19:00:00 1969
INDIRECT BLOCK=
2x INDIRECT BLOCK=
3x INDIRECT BLOCK=
F1/? for help. F2/^O for menu. Q to quit
```

Inodo: 2

La segunda entrada, es para el inode del Directorio raíz.

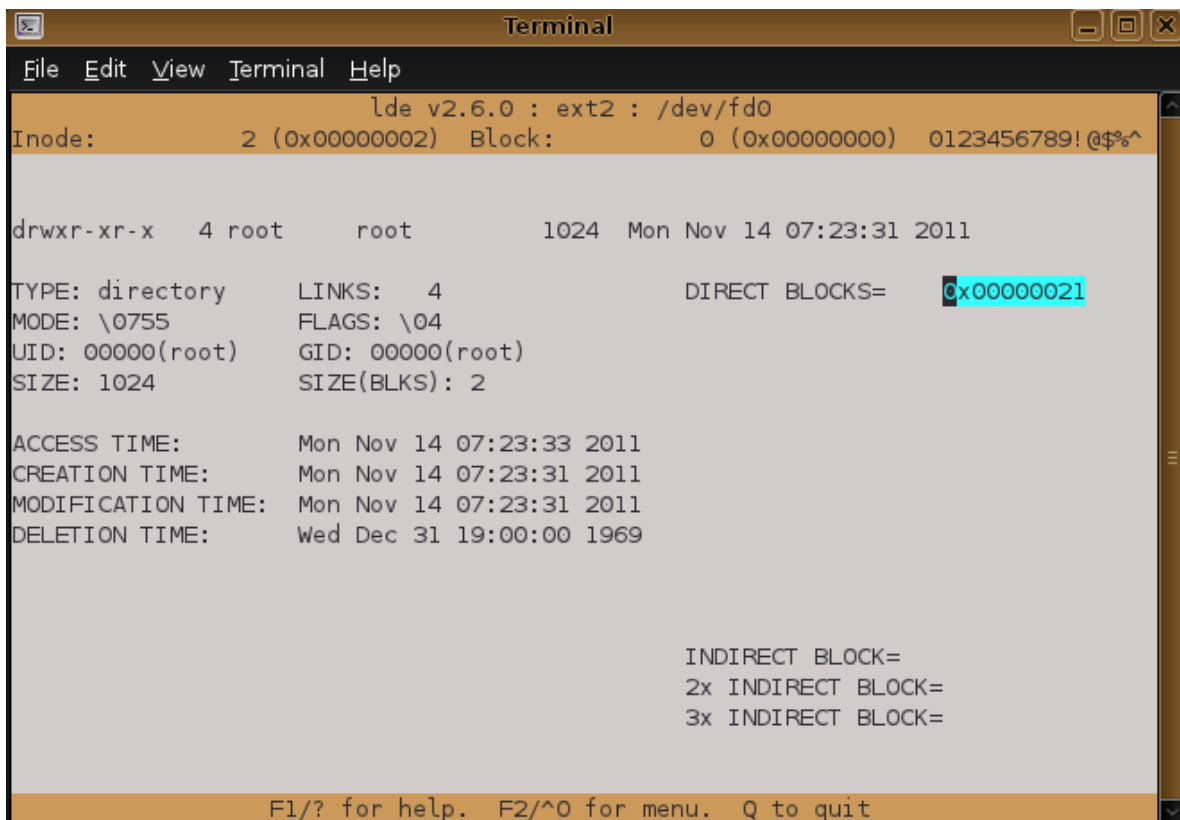
(Cuando recién ingresamos al editor, luego de ver la información del Superbloque, al presionar "I" nos ubicamos en el inode 2 correspondiente al directorio raíz)

Algunos datos que podemos ver en esta pantalla:

Tipo de Archivo (TYPE)

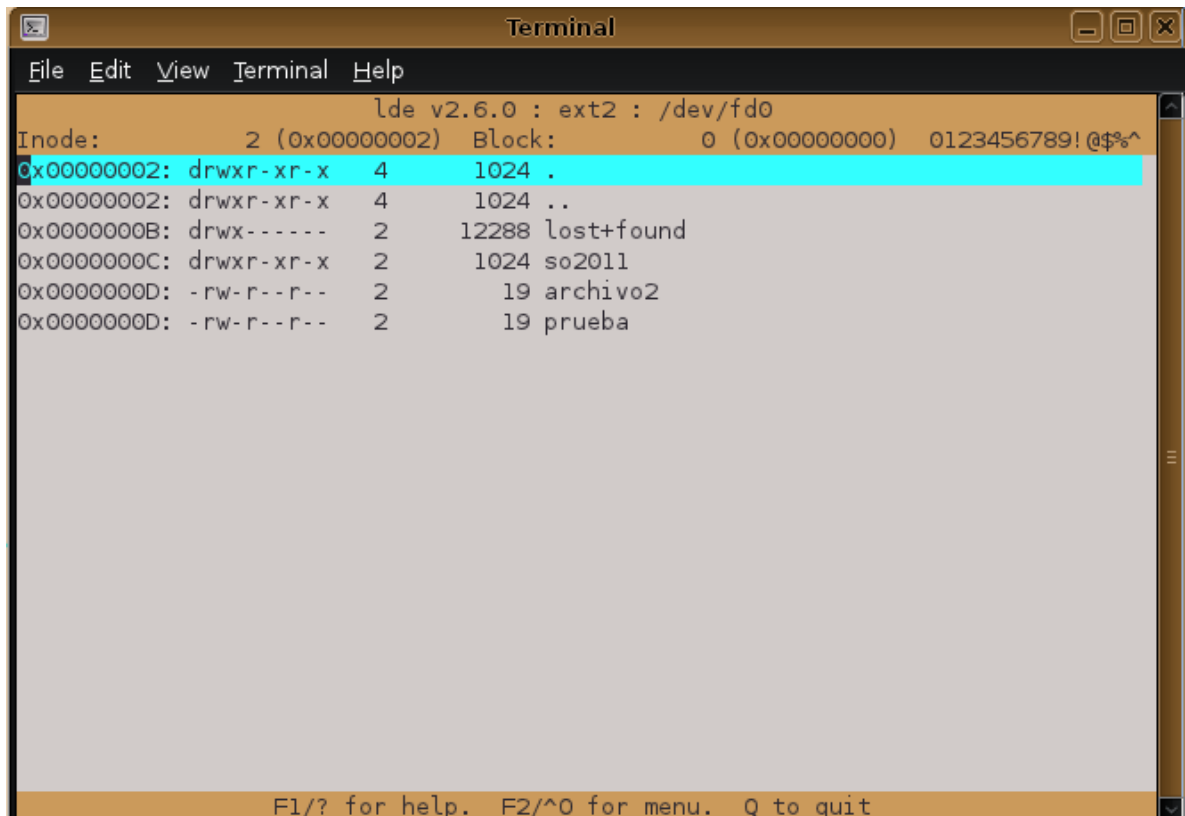
Cantidad de links al inode (4)

Punteros a bloque (Bloque directo 21)



```
Terminal
File Edit View Terminal Help
lde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002)  Block:      0 (0x00000000)  0123456789!@#$%^
drwxr-xr-x  4 root      root      1024  Mon Nov 14 07:23:31 2011
TYPE: directory    LINKS:    4          DIRECT BLOCKS= 0x00000021
MODE: \0755        FLAGS: \04
UID: 00000(root)   GID: 00000(root)
SIZE: 1024         SIZE(BLKS): 2
ACCESS TIME:       Mon Nov 14 07:23:33 2011
CREATION TIME:     Mon Nov 14 07:23:31 2011
MODIFICATION TIME: Mon Nov 14 07:23:31 2011
DELETION TIME:     Wed Dec 31 19:00:00 1969
INDIRECT BLOCK=
2x INDIRECT BLOCK=
3x INDIRECT BLOCK=
F1/? for help. F2/^O for menu. Q to quit
```

- ➔ Presionando **d** vemos los datos del inodo volcados a la salida estándar (datos binarios puro)



```
Terminal
File Edit View Terminal Help
ls -li
lsde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002) Block:      0 (0x00000000) 0123456789!@#$%^
0x00000002: drwxr-xr-x  4    1024 .
0x00000002: drwxr-xr-x  4    1024 ..
0x0000000B: drwx-----  2   12288 lost+found
0x0000000C: drwxr-xr-x  2    1024 so2011
0x0000000D: -rw-r--r--  2      19 archivo2
0x0000000D: -rw-r--r--  2      19 prueba
F1/? for help. F2/^O for menu. Q to quit
```

A partir de aquí podemos relacionar las entradas de directorio con los inodos.

➔ Desde el directorio raíz.

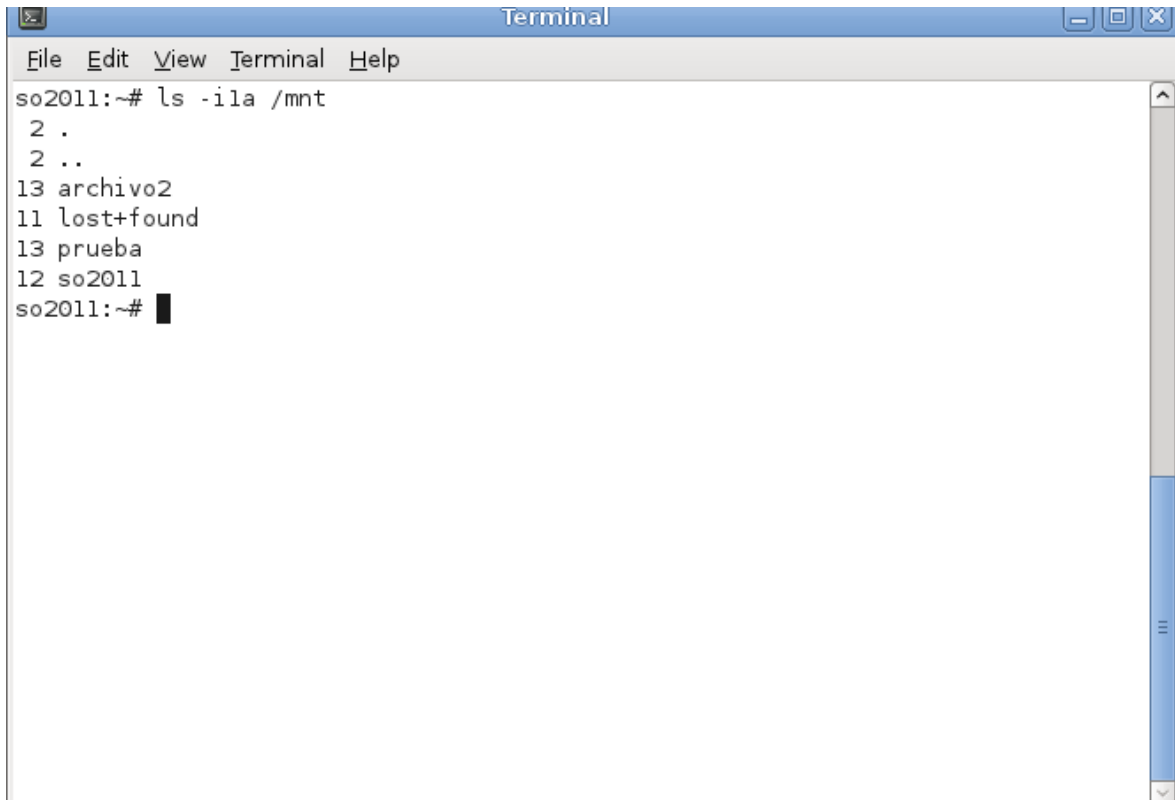
Nro de Inodo	Tamaño	Entrada de Directorio	Links a la ED	
2	1024	.	4	(el propio directorio)
2	1024	..	4	(directorio padre)
B	12288	lost+found	2	(directorio)
C	1024	so2011	2	(directorio)
D	19	archivo2	2	(dos entradas asociadas a 1 sólo inode D)
D	19	prueba	2	(el archivo prueba es un link a archivo2)

(NOTA: prueba fue creado del siguiente modo: #ln archivo2 prueba)

Verificando los datos desde línea de comandos (sin utilizar el editor):

#mount /dev/fd0 /mnt

#ls -la /mnt



```
Terminal
File Edit View Terminal Help
so2011:~# ls -la /mnt
2 .
2 ..
13 archivo2
11 lost+found
13 prueba
12 so2011
so2011:~#
```

➔ Desde el directorio lost+found.

Nro de Inodo	Tamaño	Entrada de Directorio	Links a la ED
B	12288	.	2
2	1024	..	4

➔ Desde el directorio so2011

Nro de Inodo	Tamaño	Entrada de Directorio	Links a la ED
C	1024	.	2
2	1024	..	4
E	17	archivo1	1

Volviendo a los inodos (I), se puede verificar la información en los números de inodos mencionados en las tablas.

B → Recorrido por bloques:

Bloque 0:

De acuerdo a lo que habíamos visto en el layout del disquete, el primer bloque es para el boot record. En este caso se encuentra vacío. Puede verse el mapa del bloque lleno con ceros.

```

Terminal
File Edit View Terminal Help
lde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002) Block:      0 (0x00000000) 0123456789!@#$%^
00000000 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000010 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000020 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000030 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000040 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000050 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000060 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000070 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000080 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000090 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000000A0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000000B0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
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 .....
F1/? for help. F2/^O for menu. Q to quit

```

Bloque 1:

Como cada bloque en el disquete ocupa 1024 bytes, de acuerdo al layout en el bloque identificado por 00000400 (en hexadecimal 1024, comienza el superbloque)

¿Cómo interpretamos la información?

La primera columna nos dará la posición en bytes en el dispositivo (la información está dada en hexadecimal)

Universidad Tecnológica Nacional - Facultad Regional Santa Fe

Los dos grupos de datos centrales (separados por :) representan los bytes con la información. Cada byte está representado por un par. Por ejemplo, los cuatro primeros bytes del superbloque:

B8 00 00 00 corresponden al campo: s_inodes_count

Qué representa este valor:

Debe leerse invertido ← 00 00 00 B8 (de hexadecimal) = 128 inodos (información que podemos verificar tanto en, lo arrojado por el comando para crear el fs como en la información del Superbloque vista en la segunda pantalla al ingresar al editor)

```

Terminal
File Edit View Terminal Help

lde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002) Block:      1 (0x00000001) 0123456789!@#$%^
00000400 B8 00 00 00 A0 05 00 00 : 48 00 00 00 6E 05 00 00 .....H...n...
00000410 AA 00 00 00 01 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000420 00 20 00 00 00 20 00 00 : B8 00 00 00 F0 18 C1 4E . ... ..N
00000430 F0 18 C1 4E 02 00 26 00 : 53 EF 00 00 01 00 00 00 ...N..&.S.....
00000440 00 07 C1 4E 00 4E ED 00 : 00 00 00 00 01 00 00 00 ...N.N.....
00000450 00 00 00 00 0B 00 00 00 : 80 00 00 00 38 00 00 00 .....8...
00000460 02 00 00 00 01 00 00 00 : B3 80 D3 1B 59 90 47 D2 .....Y.G.
00000470 95 D5 D9 E9 1A F3 E8 CE : 00 00 00 00 00 00 00 00 .....
00000480 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000490 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004A0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004B0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004C0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 05 00 .....
000004D0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004E0 00 00 00 00 00 00 00 00 : 00 00 00 00 7A 72 64 2B .....zrd+
000004F0 56 E2 4B 0D B9 DB 49 31 : 5D 42 33 99 01 00 00 00 V.K...I1]B3....
00000500 00 00 00 00 00 00 00 00 : 00 07 C1 4E 00 00 00 00 .....N...
00000510 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000520 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000530 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000540 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....

```

A partir de esta información, puede recorrer los bloques y verificar la información almacenada.

6) Algunas pruebas con fsck (file system check)

fsck sirve para chequear y reparar el sistema de archivo.

- 1- Corremos el **fsck** sobre el disquete:

- Primero debe estar montada la disquetera
(#mount /dev/fd0 /mnt)
- Ejecutamos y confirmamos:

```
so2011:~# fsck /dev/fd0
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/fd0 is mounted.

WARNING!!! The filesystem is mounted.  If you continue you ***WILL***
cause ***SEVERE*** filesystem damage.

Do you really want to continue (y/n)? yes

/dev/fd0 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
/dev/fd0: 14/184 files (0.0% non-contiguous), 50/1440 blocks
```

Información que rescatamos de esta corrida: de acuerdo al Paso 5 de esta guía, existen 14 inodos ocupados de 184 (files "equivalente a" inodos).

Se mantiene la información obtenida a lo largo de diferentes pasos: 184 inodos en total y 1440 bloques totales.

Si recorremos los inodos (pg_down y pg_down) encontramos:

- los inodos: 2 – 11(B) – 12(C) – 13(D) – 14(E) tienen información (ver tabla de página 13)
- mientras que, los inodos: 1 y 3 a 10(A) corresponden a inodos reservados.

Por lo tanto 14 están ocupados.

La forma de confirmar esta información es la siguiente:

- El inodo 2: corresponde al Directorio Raíz como ya se vio anteriormente.
- Para conocer cuál es el primer inodo "usable" para archivos estándares, se debe mirar el campo **s_first_ino** de la struct ext2_super_block.

Este campo se ubica en el bloque 1 (superbloque) a partir del byte 84:

El valor es: 0B 00 00 00

Se debe leer ← 00 00 00 0B (es decir a partir del inodo B(11) se puede comenzar a ocupar para archivos comunes), los anteriores están

reservados. Si volvemos a la tabla del Recorrido por Inodos (página 13) vemos que el primer inodo ocupado, luego del correspondiente al Directorio Raíz es el B(11) → Lost+found.

```

lde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002) Block:      1 (0x00000001) 0123456789!@#$%^
00000400 B8 00 00 00 A0 05 00 00 : 48 00 00 00 6E 05 00 00 .....H...n...
00000410 AA 00 00 00 01 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000420 00 20 00 00 00 20 00 00 : B8 00 00 00 F0 18 C1 4E . ... .....N
00000430 78 35 C1 4E 00 00 26 00 : 53 EF 01 00 01 00 00 00 x5.N..&.S.....
00000440 78 35 C1 4E 00 00 4E ED 00 : 00 00 00 00 01 00 00 00 x5.N.N.....
00000450 00 00 00 00 0B 00 00 00 : 80 00 00 00 38 00 00 00 .....8...
00000460 02 00 00 00 01 00 00 00 : B3 80 D3 1B 59 90 47 D2 .....Y.G.
00000470 95 D5 D9 E9 1A F3 E8 CE : 00 00 00 00 00 00 00 00 .....
00000480 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000490 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004A0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004B0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004C0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 05 00 .....
000004D0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000004E0 00 00 00 00 00 00 00 00 : 00 00 00 00 7A 72 64 2B .....zrd+
000004F0 56 E2 4B 0D B9 DB 49 31 : 5D 42 33 99 01 00 00 00 V.K...I1]B3....
00000500 00 00 00 00 00 00 00 00 : 00 07 C1 4E 00 00 00 00 .....N...
00000510 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000520 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000530 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000540 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
Enter block number (leading 0x or $ indicates hex): > , ^ [[M 37^ [[M#37

```

2- Se modificará manualmente el FS con el editor:

- Arbitrariamente se decide modificar el contenido del "Inode bitmap".

¿Qué información encontramos en el "Inode bitmap"

La cantidad de inodos ocupados (esto se representa con 1 bit, es decir la presencia de un 1 indica inodo ocupado).

- ¿Cómo llegamos al "inode bitmap"?

La dirección del primer bloque, donde comienza el "inode bitmap" está en el campo: **bg_inode_bitmap** de la **struct ext2_group_desc**.

Con esa estructura, representamos el "Group Descriptors", que de acuerdo al layout del disquete se ubica en el bloque seguido al superbloque. Es decir en el bloque 2 de offset 2048 (en hexadecimal 800).

El campo que estamos buscando, según la definición en la estructura, es el segundo. Debemos correrlos 4 bytes (ya que el primer campo corresponde al bg_block_bitmap y es de ese tamaño). Entonces:

bg_inode_bitmap = 09 00 00 00 (leemos invertido ←)

El bitmap de inodes está ubicado en el bloque 9

```
lde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002) Block:      2 (0x00000002) 0123456789!@#$%^
00000800 08 00 00 00 09 00 00 00 : 0A 00 00 00 6E 05 AA 00 .....n...
00000810 03 00 04 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000820 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000830 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000840 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000850 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000860 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000870 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000880 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000890 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000008A0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000008B0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000008C0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000008D0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000008E0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
000008F0 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000900 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000910 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000920 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000930 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
00000940 00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00 .....
Block (2) copied into copy buffer.
```

Presionando # podemos colocar el número de bloque y movernos hasta ahí (en este caso el bloque 9).

```

lde v2.6.0 : ext2 : /dev/fd0
Inode:      2 (0x00000002) Block:      9 (0x00000009) 012-456789!@$%^
00002400  FF 3F 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ?.....
00002410  00 00 00 00 00 00 00 FF : FF FF FF FF FF FF FF FF  .....
00002420  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002430  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002440  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002450  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002460  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002470  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002480  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002490  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
000024A0  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
000024B0  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
000024C0  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
000024D0  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
000024E0  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
000024F0  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002500  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002510  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002520  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002530  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....
00002540  FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF  .....

```

Cada bit representará la presencia de un inodo, si miramos en esta pantalla tenemos la siguiente información:

FF 3F → igual a 14 (cantidad de inodos ocupados)

Esto es:

F				F				3				F			
2 ³	2 ²	2 ¹	2 ⁰	2 ³	2 ²	2 ¹	2 ⁰	2 ³	2 ²	2 ¹	2 ⁰	2 ³	2 ²	2 ¹	2 ⁰
1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1
1 byte= 8 bits								6 bits							

El total será, 14 bits = 14 inodos ocupados.

A continuación siguen bytes en cero. Si contamos el total de estos bytes en cero, el valor será igual a 21. Si lo llevamos a bits (cada par de ceros = 8 bits), tendremos 168 bits (representan cantidad de inodos libres).

Sumando los inodos ocupados (**14**) + los inodos libres (**170** = 168 en grupos de 0 + los 2 bits que están libres en el segundo byte -3F-) obtenemos el total de **184 inodos**.

Modificamos entonces la cantidad de inodos (vamos a restarle uno, esto implicará que el FS quede inconsistente):

1. Con **F** ingrese a las flags del editor.
2. Modifique para que puede escribir el FS (presione **w**, verá que en la línea correspondiente a "OK to write to FS" → FW toma el valor "yes". Con **q** vuelve a la edición).
3. En el byte que tiene los valores **3F** modificamos a **2F**. (Colocamos E – sobre el 3 y luego 2F. Con la flecha nos movemos al siguiente byte, se verán los valores editados).
4. Con **Q** salimos del editor (previa confirmación de la edición).

```

Terminal
File Edit View Terminal Help
lde v2.6.0 : ext2 : /dev/fd0
Inode: 2 (0x00000002) Block: 9 (0x00000009) 01-3456789! @3%^
00002400 FF 2F 00 00 00 00 00 : 00 00 00 00 00 00 00 00 ./
00002410 00 00 00 00 00 00 00 FF : FF FF FF FF FF FF FF FF
00002420 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002430 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002440 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002450 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002460 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002470 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002480 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002490 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
000024A0 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
000024B0 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
000024C0 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
000024D0 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
000024E0 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
000024F0 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002500 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002510 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002520 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002530 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF
00002540 FF FF FF FF FF FF FF FF : FF FF FF FF FF FF FF FF

```

- Corremos ahora el fsck sobre el disquete con el FS modificado:

Nos informa sobre una diferencia en el mapa de bits del nodo i: ... si quiere corregirlo, confirmamos.

La salida es la siguiente:

```
so2011:~# fsck /dev/fd0
fsck from util-linux-ng 2.17.2
e2fsck 1.41.12 (17-May-2010)
/dev/fd0 was not cleanly unmounted, check forced.
Pass 1: Checking inodes, blocks, and sizes
Pass 2: Checking directory structure
Pass 3: Checking directory connectivity
Pass 4: Checking reference counts
Pass 5: Checking group summary information
Inode bitmap differences: +13
Fix<y>? yes

/dev/fd0: ***** FILE SYSTEM WAS MODIFIED *****
/dev/fd0: 14/184 files (0.0% non-contiguous), 50/1440 blocks
```

Si Ingresa nuevamente al editor y va al bloque 9, podrá ver que el Inode Bitmap tiene la siguiente información FF 3F 00 .. (es decir si tiene los 14 inodos contados como ocupados: por lo que el FS vuelve a ser consistente).