Ext2 y Ext3

Las estructuras de los filesystems ext2 y ext3 son prácticamente las mismas, lo que cambia es que ext3 agrega el llamado "**journaling**" (**ver al final la definición*).

Siempre que se mencione "**bloque**", hace referencia a un bloque lógico, es decir, a un **'cluster'**. Por lo que prácticamente todo el filesystem se estructura en clusters. Por ejemplo, se asigna un cluster entero para el área de boot, otro para el superbloque, otro para los bitmaps, etc… (por más que luego quede espacio libre, ya que recordemos que el tamaño del cluster se define al momento de darle formato a la partición; y este queda guardado en el superbloque).

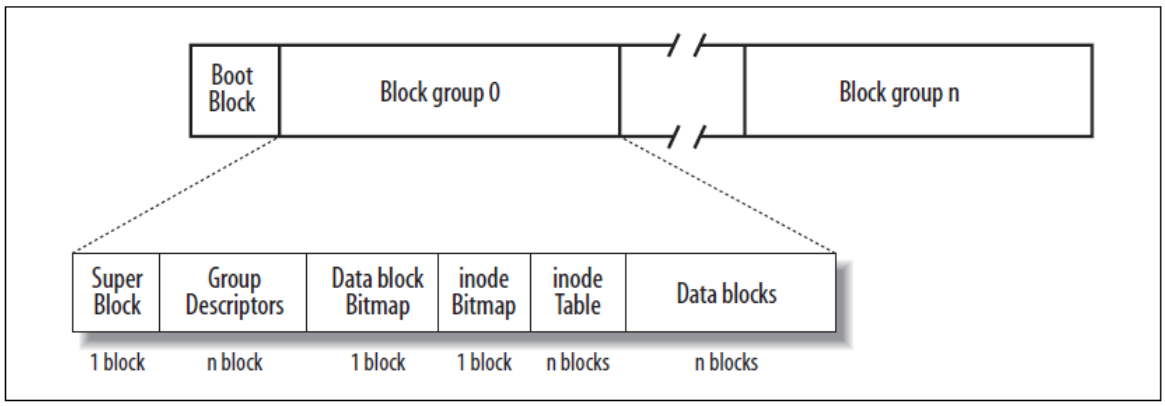


Figure 17-1. Layouts of an Ext2 partition and of an Ext2 block group

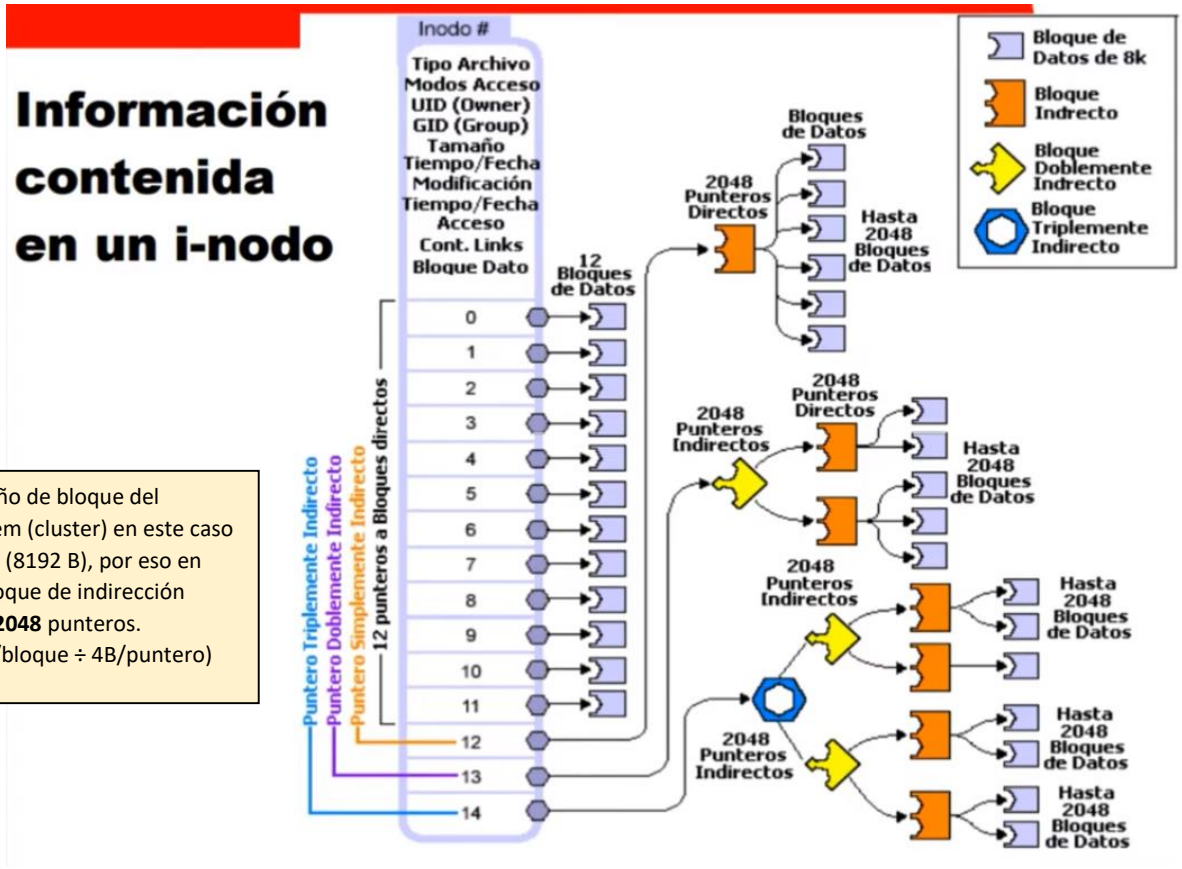
Por lo dicho anteriormente, también se puede determinar por ejemplo cuántas entradas tendrá la tabla de inodos, ya que el bitmap (secuencia de bits que determinan ocupado/libre) de inodos debe entrar en un bloque, y si el tamaño es de por ejemplo 4K (4096 Bytes), entonces habrá 32.768 inodos. Y cada inodo (por definición) ocupa 128 Bytes, y en este ejemplo en un bloque entrarían 32 inodos. Entonces, la tabla de inodos ocupará 32.768/32 = **1024 bloques**, donde cada bloque tendrá 32 entradas y en total son 32.768. (Nota: *esto sería para un solo 'block group', pero es lo mismo para los demás, y así podríamos calcular los números totales de todo el filesystem*).

i-nodos

La forma de hacer referencia a los archivos y directorios del filesystem, es mediante los '**inodos**'. El filesystem tiene tablas de inodos donde justamente cada entrada/registro es un inodo.
Un inodo es una estructura que contiene por un lado **metadatos** y por otro, **punteros** a los bloques de datos del archivo o directorio. Tiene 12 punteros directos a los bloques de datos y 3 punteros indirectos, uno simple (apunta a un bloque que tiene 'n' punteros directos), uno doble (apunta a un bloque que tiene 'n' punteros indirectos simples), y uno triple (apunta a un bloque que tiene 'n' punteros indirectos dobles).

El 'n' queda determinado por el tamaño del bloque dividido el tamaño de un tipo puntero. Por ejemplo, si los bloques son de 4k (4096 B), y los punteros ocupan 4 Bytes -> n = 1.024 punteros. (¿Cuál es el verdadero tamaño de los punteros en ext2/3? → **4 Bytes**).

PREGUNTAR SI ESTO ES ASI (SI LOS PUNTEROS INDIRECTOS APUNTAN A BLOQUES COMO TAL RELLENOS DE SOLO PUNTEROS, O APUNTAN A ESTRUCTURAS CON SOLO 12 PUNTEROS). **Sí, es así. Los punteros indirectos apuntan a bloques "de indireccion", es decir, bloques que únicamente tienen punteros de 4 Bytes en su interior, tantos como entren**.



El tamaño de bloque del filesystem (cluster) en este caso sería **8K** (8192 B), por eso en cada bloque de indirección entran **2048** punteros. (8192B/bloque ÷ 4B/puntero)

https://www.youtube.com/watch?v=izsYILDYW-A

Esta estructura de inodos, permite que podamos posicionarnos en cualquier bloque de datos de un archivo del filesystem en a lo sumo **4 accesos directos** (3 indirecciones y la lectura del bloque; *inodo_entry_pointer*[14] → *indireccion_3*[x] → *indireccion_2*[y] → read(*indireccion_1*[z])).

Aunque realmente los bloques de indirección creo que habría que recorrerlos secuencialmente.

Nota: Cada estructura de inodo se asocia con un **único** archivo/directorio; no puede contener metadatos o punteros referidos a más de 1 archivo. Pero varios archivos pueden tener una referencia al mismo inodo (pudiendose así tener 'accesos directos': **hards links [**softs no sé si se pueden → **Sí, también** (y también se conocen como *'symbolic links'*)**] *ver imagen al final**).

Esto es así porque dentro del filesystem tendremos una determinada cantidad de inodos, cada uno con un **identificador** único, y cada archivo tendrá como metadato el identificador del inodo que lo representa. Además, como metadato también tiene su propio nombre de archivo; no está dentro de los metadatos del inodo (EMMM, ESTOS 2 METADATOS QuE MENCIONÉ CREO QuE SE GUARDAN MAS BIEN EN EL BLOQUE DE DATOS ASOCIADO AL DIRECTORIO QUE 'CONTENGA' AL ARCHIVO. **En efecto, es así**. Los bloques de datos de los archivos, tienen solo datos. El bloque de datos del directorio que "contiene" a un archivo/directorio, es el que tiene los metadatos: nombre y n° de inodo. Estos bloques de directorio, en sí, contienen '**directory entries**').

Directorios

Los bloques de datos de los directorios (recordemos que también son archivos), en vez de contener datos como tal, contienen metadatos sobre los archivos o directorios que hace referencia. Por ejemplo, tienen por cada entrada (cada 'registro' asociado a un archivo) el identificador del inodo que hace referencia al archivo (son como los 'directory entry' de FAT32, pero con otros atributos/campos y además **son de tamaño variable**).
Recordemos además, que cada directorio está asociado a un inodo, por lo que estos bloques de (meta)datos de los directorios, son referenciados justamente por un inodo.

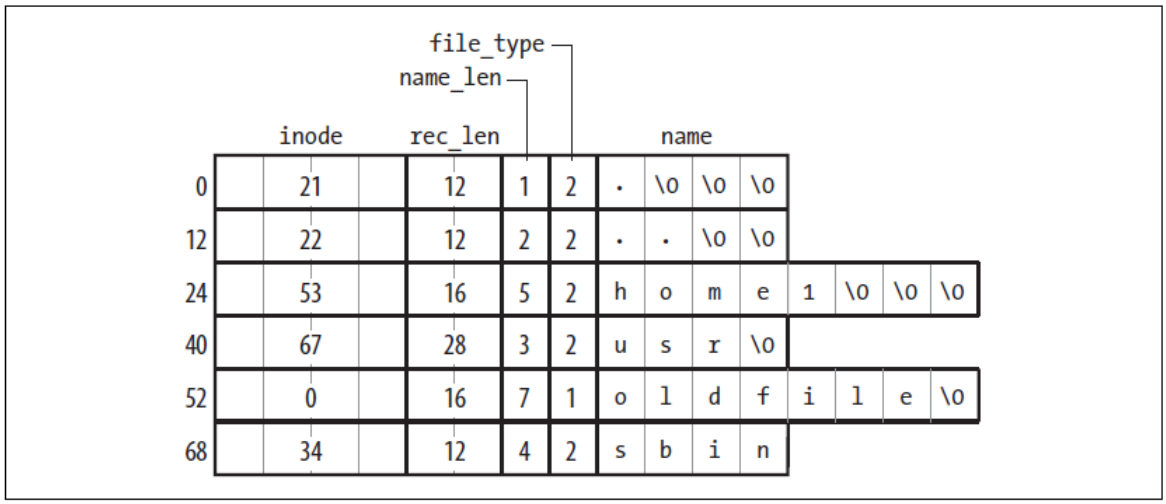
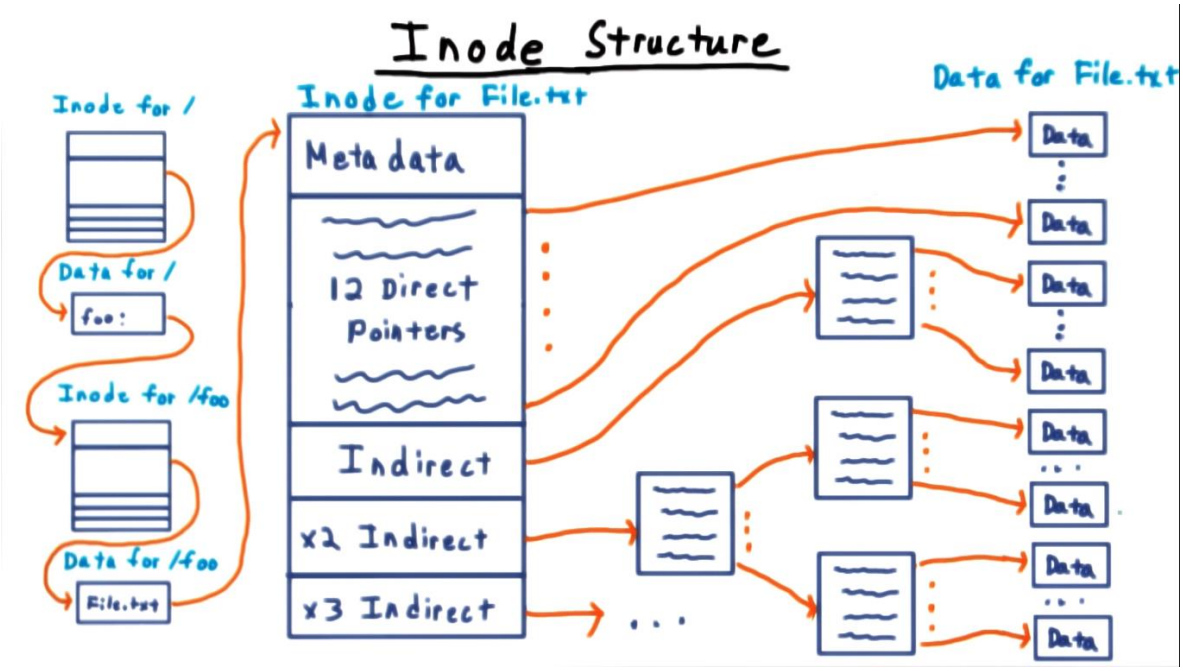


Figure 17-2. An example of the EXT2 directory

Nota: lo que no se bien es cómo hace el mapeo del número de inodo, porque si por ejemplo se pasa del límite de la tabla de inodos del 'block group' actual (todos los inodos están ocupados y se crea un archivo dentro de un directorio del 'block group' actual), lo que significa que pertenece a otro 'block group', ¿cómo hace? Quizás se lea el siguiente 'block group' y se resta el número del inodo con la longitud de la tabla de inodos anterior.

Ejemplo de un directorio que contiene (*hace referencia a*) un archivo:



https://www.youtube.com/watch?v=tMVj22EWg6A

Nota: *el directorio '**/**' es el directorio **raíz** (así se representa en los filesystems ext)*.

A grandes rasgos podríamos decir que: *(→: tiene/hace_referencia_a ó "linkea" a)*

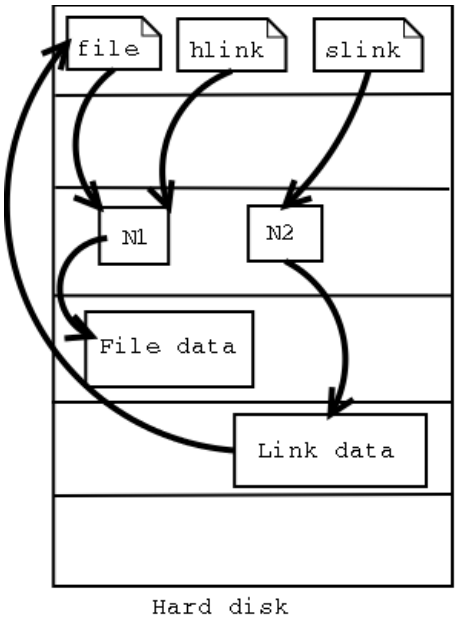
Filesystem ext2/3 → inodos de directorios → nombres de archivos → inodos de archivos → datos

El filesystem tiene referencias de inodos de archivos e inodos de directorios. Los inodos de archivos tienen la referencia a los datos en sí. Los inodos de directorios tienen la referencia de nombres de archivos o directorios, y estos al inodo que los representa. Para acceder a los datos de un archivo, nos vamos moviendo por los inodos de los directorios que lo contienen (empezando por el directorio raíz), hasta llegar al inodo del archivo que tiene las referencias a los bloques de datos (este recorrido sería el 'path', que se va formando gracias a los 2 directorios 'ocultos' que contiene cada directorio: uno con la referencia del inodo actual '**.**' y otro con la del padre '**..**').

Extras

1. El '**journaling**' es un log/bitácora de las escrituras/write's que están planeadas hacerse en archivos; ya que a veces (**o siempre???**) realmente se escribe en disco cuando desmontamos correctamente una partición, y si antes de eso, o justo durante el proceso de escritura, por ejemplo se nos apaga la pc **abruptamente** por 'x' motivo, luego se checkea el journaling para ver qué escrituras llegaron a hacerse, y cuáles no (en vez de tener que revisar todo el disco en busca de fallos). Entre las que no, en general podemos elegir si continuar con la escritura, o cancelarla.

2. Estructura de los ***hard links*** y ***soft links***.



https://tldp.org/LDP/intro-linux/html/sect_03_03.html