Supóngase un sistema de ficheros UNIX estándar con las siguientes características:

- i-nodos de 56 bytes (para descripción y atributos del fichero) más 6 punteros directos, un indirecto simple, otro doble y otro triple.
- Tamaños de los punteros a i-nodos y bloques: 64bits y tamaño de bloque de 16KiB.
- Directorio con entradas estáticas de tamaño estático (116 bytes).

a) **[1 punto]** ¿Cuál es el tamaño del fichero más grande que se puede almacenar? Indique dicho límite según dos restricciones:
   i. El número de bloque más alto que se puede direccionar con un puntero de bloque.
   ii. Los niveles de indirección disponibles desde un i-nodo.

b) **[1 punto]** ¿Cuántos i-nodos entran en un bloque de i-nodos?

c) **[1 punto]** ¿Cuántos bloques de datos ocuparía un directorio con 1024 entradas?

Si se tiene un sistema de ficheros de 2TB cuyo contenido es:

DIRECTORIO RAÍZ	
.	
..	
DIR1	
IMAGEN.jpg	128KiB

DIRECTORIO DIR1	
.	
..	
DATA.raw	512GiB
FICHERO.txt	4KiB

d) **[2 puntos]** Indique el número de i-nodos, bloques de indirección y bloques de datos que ocupan cada uno de los siguientes elementos: DIR1, IMAGE.jpg, DATA.raw y FICHERO.txt

e) **[2 puntos]** Si ahora se creasen los siguientes enlaces ¿Cuántos i-nodos y bloques de datos/indirección habría que utilizar? Indique exactamente cómo se modificarían los i-nodos existentes (si hiciese falta):
   i. Un enlace físico llamado DIR1/ENLACE1 que apuntase al fichero IMAGEN.jpg
   ii. Un enlace simbólico llamado ENLACE2 que apuntase al fichero DIR1/DATA.raw

En muchos sistemas de ficheros se dispone de funcionalidades para manejar ficheros dispersos (*sparse files*) que son ficheros (por lo general de gran tamaño) pero que tienen porciones muy grandes de los mismos llenas de bytes a cero.

Una de las formas de gestionar estos ficheros es haciendo que los bloques de datos llenos de bytes a cero no se reserven y que, de la misma manera, si un bloque de indirección tiene sólo bloques con bytes a cero, tampoco se crea ese nivel de indirección.

Si, en nuestro caso, supóngase que el fichero DIR1/DATA.raw está íntegramente compuesto de bytes a cero salvo los primeros 16 bytes, 128 bytes más en la posición 2^{30}, y los últimos 64 bytes del fichero.

f) **[1 punto]** Dibuje el árbol de indirección del fichero DIR1/DATA.raw, indicando cuántos bloques de datos y bloques de indirección están ocupados.

Si ahora se hiciese la siguiente operación (supóngase que **`buff`** es un buffer de 24KiB con un contenido distinto de cero):

```
fd=open("/DIR1/DATA.raw", O_RDWR);
lseek(fd, 1024*1024, SEEK_SET);
write(fd, buff, 24*1024);
```

g) **[2 puntos]** Asumiendo que el fichero no se ha usado previamente. Detalle, paso a paso, que operaciones de lectura y escritura de bloque se realizarán para:
   i. Ejecutar la llamada *open()*, que implica resolver la ruta y cargar el i-nodo.
   ii. Y ejecutar las operaciones *lseek()*/*write()* posteriores.

SOLUCIÓN

a) **[1 punto]** ¿Cuál es el tamaño del fichero más grande que se puede almacenar? Indique dicho límite según dos restricciones:
   i. El número de bloque más alto que se puede direccionar con un puntero de bloque.
   ii. Los niveles de indirección disponibles desde un i-nodo.

De acuerdo al puntero de bloque, que es de 64 bits, el número de bloque más alto es el 2^{64} como cada bloque es de 16KiB tenemos:

2^{64} bloques x 16KiB/bloque = 2^{64} x 2^{14} B = 2^{78} B → 2^{18} EiB [Exabytes] o lo que es lo mismo 1/4 YiB [YottaByte]

Atendiendo a lo que son niveles de indirección:

- Número de bloques direccionables desde un bloque de indirección (direcciones de 64 bits → 8 bytes): 16KiB/bloque / 8bytes/dirección = 2^{14} / 2^{3} direcciones = 2^{11} direcciones = 2048 direcciones .
- Desde un i-nodo se direccionan 6 punteros directos, un indirecto simple, otro doble y otro triple:
  - Directos: 6 x 16KiB
  - Simple: 1 x 2^{11} x 16KiB
  - Doble: 1 x 2^{11} x 2^{11} x 16KiB
  - Triple: 1 x 2^{11} x 2^{11} x 2^{11} x 16KiB
- En total estamos hablando de un fichero del orden de 2^{11} x 2^{11} x 2^{11} x 16KiB = 2^{33} x 2^{14} B= 2^{47} B → 128 TiB [Terabytes].

b) **[1 punto]** ¿Cuántos i-nodos entran en un bloque de i-nodos?

Tamaño del i-nodo 56 bytes + (6+1+1+1) punteros x 8 bytes/puntero=128 bytes.
Bloque 16KiB / 128 bytes/i-nodo = $2^{14}/2^{7}$ i-nodos = 128 i-nodos.

c) **[1 punto]** ¿Cuántos bloques de datos ocuparía un directorio con 1024 entradas?

Entrada de directorio 116 bytes + puntero de i-nodos = 124 bytes ≈ 128 bytes = 2^{7} bytes

Un directorio con 1024 entradas → 2^{10} entradas x 2^{7} bytes/entrada = 2^{17} bytes

En bloques: 2^{17} bytes / 16KiB/bloque = $2^{17}/2^{14}$ bloques = 8 bloques (es decir que ocuparía los 6 bloques directos, 1 bloque indirecto y dos entradas de este a otros dos bloques.

d) **[2 puntos]** Indique el número de i-nodos, bloques de indirección y bloques de datos que ocupan cada uno de los siguientes elementos: DIR1, IMAGE.jpg, DATA.raw y FICHERO.txt

Elemento	i-nodos	Bloques datos	Bloques indirección
DIR1 (4 entradas)	1	1 (menos de 16 KiB)	0
IMAGE.jpg (128KiB)	1	128KiB/16KiB= 8	1 (directo)
DATA.raw (512GiB)	1	512 GiB / 16 KiB = 2^{39} B / 2^{14}B = 2^{25}	1 (directo: permite 2^{11})+ 1+2^{11} (doble: permite 2^{22})+ 1+(7x2^{11}) = 3 + 2^{14}
FICHERO.txt (4KiB)	1	1 (menos de 16 KiB)	0

e) **[2 puntos]** Si ahora se creasen los siguientes enlaces ¿Cuántos i-nodos y bloques de datos/indirección habría que utilizar? Indique exactamente cómo se modificarían los i-nodos existentes (si hiciese falta):
   i. Un enlace físico llamado DIR1/ENLACE1 que apuntase al fichero IMAGEN.jpg
   ii. Un enlace simbólico llamado ENLACE2 que apuntase al fichero DIR1/DATA.raw

Enlace físico: No se crea ningún i-nodo, se modifica en el i-nodo existente el número de enlaces.

Enlace simbólico: Se crea un nuevo i-nodo y un nuevo bloque de dato que contiene la ruta al fichero enlazado.

f) **[1 punto]** Dibuje el árbol de indirección del fichero DIR1/DATA.raw, indicando cuántos bloques de datos y bloques de indirección están ocupados.

El fichero tiene sólo 3 fragmentos de pocos bytes con datos (distintos de cero), el comienzo, una posición intermedia (exactamente la posición 2^{30}) y el final del archivo. Eso implica que, de forma efectiva, sólo habrá 3 bloques de datos ocupados, y los necesarios de indirección a los mismos.

¿Dónde se encuentran estos bloques?

- Los primeros 16 bytes: Evidentemente, el primer bloque de disco referido desde el i-nodo directamente.
- 128 bytes más en la posición 2^{30}: La posición 2^{30} se corresponde con el bloque: $2^{30}/2^{14}=2^{16}$ que está en el rango del puntero doble. Por lo tanto habrá un puntero doble, y un puntero simple asociado a éste ocupados adicionalmente.
- Los últimos 64 bytes del fichero: Que están en el rango del puntero triple, así que ese bloque triple, un doble y un indirecto simple también estarían rellenos.

g) **[2 puntos]** Asumiendo que el fichero no se ha usado previamente. Detalle, paso a paso, que operaciones de lectura y escritura de bloque se realizarán para:
   i. Ejecutar la llamada open(), que implica resolver la ruta y cargar el i-nodo.
   ii. Y ejecutar las operaciones lseek()/write() posteriores.

La operación implica escribir 24KiB en la posición 2^{20} (que es el bloque $2^{20}/2^{14}=2^{6}$, dentro del rango del puntero simple)

Open(): Hay que resolver la ruta lo cual implica:

- Acceder al i-nodo del dir. Raíz.
- Acceder al bloque de contenido del directorio raíz (buscar la entrada DIR1).
- Acceder al i-nodo del dir. DIR1
- Acceder al bloque de contenido del directorio DIR1 (buscar la entrada DATA.raw)
- Acceder al i-nodo del fichero DATA.raw

Lseek()/Write():

- El acceso al puntero simple, como no existe (está a cero), se crea.
- Se crea un bloque a ceros y se referencia desde el bloque indirecto simple.
- Ídem para el segundo bloque.
- Se escribe el bloque en disco (de forma diferida).