Dada la siguiente configuración de base del sistema de ficheros: Un sistema de ficheros de 32TiB, con i-nodos de 512 bytes (con 8 entradas de punteros directos, uno indirecto simple, otro doble y un triple, y un campo de 6 bytes para indicar la longitud del fichero), direcciones de bloque y de i-nodo de 64 bits, tamaño del bloque lógico 8KiB y formato de las entradas de directorio (i-nodo, nombre de la entrada [504 caracteres de 8 bits]).

a) **[1,0/10]** Indique cuál es el límite del tamaño de fichero más grande que se puede almacenar:
   - Debido al tamaño del sistema de ficheros.
   - Debido al número de bloque más alto que se puede numerar.
   - Debido al máximo nivel de indirección desde el i-nodo.
   - Debido a las limitaciones para indicar en el i-nodo la longitud del fichero.

   Importante: indique los resultados **en bytes** y en **la potencia de 2 más próxima**.

Se desea evaluar las estrategias de almacenamiento para un sistema inteligente de cámaras de vigilancia. El sistema está compuesto por un centenar de cámaras que recogen información de forma continua y la vuelcan con una frecuencia de horas a un sistema de ficheros para su almacenamiento. Cada volcado implica crear un directorio con la siguiente estructura:

- *Video_seq.raw*: Secuencia de video correspondiente a ese rango de fechas, son ficheros de unos 8GiB de tamaño.
- *Timestamp.txt*: Rango de fecha/hora de comienzo y final de esa secuencia ocupa 32bytes.
- *CameraID.txt*: Un enlace al fichero que contiene información de la cámara (número de serie, localización, etc..). Serían referencias a ficheros del tipo "../../cameras/RQ18-23P.txt".
- *Previous_seq*: Un enlace al directorio que contiene la secuencia de video anterior de esa misma cámara. Aquí se trataría de una referencia del tipo "../044522".
- *Keyframes.csv*: Fichero de texto separado por comas que contiene la referencia de los fotogramas donde se detecta movimiento en la secuencia de vídeo. El tamaño típico son unos 128KiB.

b) **[1,0/10]** De las dos entradas que son enlaces (*CameraID.txt* y *Previous_seq*) ¿qué tipo o tipos de enlace se podría aplicar en cada caso? Si se quisiese minimizar el consumo de espacio, ¿cuál sería la opción de diseño a tomar?

c) **[0,5/10]** Si el directorio "cameras" donde están los ficheros con información de las cámaras se encontrase montado en un sistema de ficheros diferente, ¿afectaría eso a los tipos de enlaces que se puede establecer?

d) **[1,0/10]** Para los datos del sistema de ficheros antes indicado, y las opciones de diseño tomadas en el apartado b), indique tanto para el propio directorio, como para las entradas y ficheros que contiene:
   - Número de i-nodos que se ocupan.
   - Número de bloques indirectos ocupados.
   - Número de bloques de datos

   Utilice los valores de tamaño típico de estos ficheros para hacer los cálculos.

e) **[1,5/10]** Dibuje una representación gráfica de i-nodos y bloques para ese directorio y su contenido, indicando entradas de directorio y números de bloque e i-nodo, considerando:
   - El primer i-nodo libre es el 5012 (de ahí en adelante todos están libres).
   - El primer bloque libre es el 7321 (de ahí en adelante todos están libres).
   - El enlace *CameraID.txt* apunta al fichero "../../cameras/RQ18-23P.txt" que está en el i-nodo 1102 y su contenido comienza en el bloque 2911.
   - El enlace *Previous_seq* apunta al directorio "../044522" que está en el i-nodo 3721 y su contenido comienza en el bloque 3309.

Si el sistema de ficheros también tiene una funcionalidad específica para optimizar el almacenamiento de los ficheros de pequeño tamaño, sustituyendo los punteros a bloque del i-nodo por el contenido del fichero, indique:

f) **[0,5/10]**¿Hasta qué el tamaño de fichero podría almacenarse dentro del i-nodo?

g) **[0,5/10]** ¿A cuáles de las entradas del directorio se le podría aplicar esta funcionalidad? ¿Y al propio directorio?

Supongamos que ahora el sistema de ficheros además dispone de soporte para *extents*, con el siguiente formato:

- 2 bits para indicar que en lugar de un puntero a bloque es un *extent*.
- 48 bits para indicar la posición en bloques dentro de ese fichero donde comienza el *extent*.
- 64 bits para indicar el primer bloque de los datos del *extent*.
- 14 bits para indicar el número de bloques que componen ese *extent*.

h) **[0,5/10]** ¿Qué tamaño puede direccionar un único *extent*? ¿Y un bloque indirecto completo de *extents*?

i) **[2,0/10]** Usando esta funcionalidad, ¿cuántos bloques ocuparía el fichero *Video_seq.raw*? Además, dibuje la estructura de i-nodos y bloques de este fichero.

j) **[1,5/10]** Suponga que se almacena un fichero verdaderamente grande, del orden de varios TiB de datos. Si se quiere acceder a una posición concreta justo en la mitad del fichero (como podría ser el caso de la búsqueda de instantes de determinados en las secuencias de vídeo), ¿que ventaja o inconveniente tiene el uso de *extents* para realizar esa indexación directa?. Tenga en cuenta que los *extents* son espacios contiguos internamente, pero no necesariamente dos *extents* tiene que estar contiguos entre sí. ¿Es necesario complementar alguno de estos mecanismos con alguna estructura de apoyo?

Administración:

Asocie los siguientes términos con la funcionalidad de administración de sistemas para la que se aplican:

iptables		Servidor de nombres dentro del servicio Samba.
inetd		Configuración de filtros Firewall, que se aplican dentro de la pila de protocolos.
nmbd		Configuración del límite de recursos de sistema por usuario.
ulimit		Servidor que atiende a múltiples servicios básicos de red (ni *standalone* ni RPC).
crond		Servidor asociado a la configuración de tablas de tareas programadas.

Solución:

a) **[1,0/10]** Indique cuál es el límite del tamaño de fichero más grande que se puede almacenar:
   - Debido al tamaño del sistema de ficheros: 32TiB = 2^{45} bytes
   - Debido al número de bloque más alto que se puede numerar: 2^{64} bloques x 8 KiB/bloque = 2^{64} x 2^{13} bytes = 2^{77} bytes
   - Debido al máximo nivel de indirección desde el i-nodo: Será el asociado al indirecto triple, para ello debemos calcular le número de bloques direccionables por un bloque indirecto (8KiB/bloque / 64 bits/dir-bloque)=$2^{13}/2^3$ dir-bloque/bloques=2^{10} direcciones de bloque por bloque). Como hay 3 niveles de indirección se podrán direccionar $(2^{10})^3$ bloques de 8 KiB/bloque = 2^{30} x 2^{13} bytes = 2^{43} bytes (los bloques dobles y simples aportan poco más de espacio).
   - Debido a las limitaciones para indicar en el i-nodo la longitud del fichero: Como se usan 6 bytes (48 bits), y direccionado el tamaño del fichero a nivel de byte la última posición direccionable sería 2^{48} bytes.

b) **[1,0/10]** De las dos entradas que son enlaces (*CameraID.txt* y *Previous_seq*) ¿qué tipo o tipos de enlace se podría aplicar en cada caso? Si se quisiese minimizar el consumo de espacio, ¿cuál sería la opción de diseño a tomar?:
   - *CanalID.txt*: Como es un fichero podrá usar tanto enlaces físicos como simbólicos. El más eficiente en consumo de recursos es un físico (ya que no usa ni un i-nodo nuevo ni un bloque de datos).
   - *Previous_seq*: Como referencia a un directorio sólo puede ser enlazado por medio de un enlace simbólico.

c) **[0,5/10]** Si el directorio "cameras" donde están los ficheros con información de las cámaras se encontrase montado en un sistema de ficheros diferente, ¿afectaría eso a los tipos de enlaces que se puede establecer?: Sí, un enlace físico no se puede establecer entre volúmenes/sistemas de ficheros diferentes. Ambos enlaces tendrían que ser, por tanto simbólicos.

d) **[1,0/10]** Para los datos del sistema de ficheros antes indicado, y las opciones de diseño tomadas en el apartado b), indique tanto para el propio directorio, como para las entradas y ficheros que contiene:

   Resulta necesario calcular el tamaño real de determinados elementos:
   - El directorio contiene 4 entradas + 2 entradas "." y "..", 6 entradas en total, cada entrada tiene un número de i-nodo (64 bits) y 504 caracteres (504 bytes): 8 bytes + 504 bytes= 512bytes. 6 entradas x 512 bytes por entrada=3 KiB
   - El fichero *Video_seq.raw* que ocupa 8GiB: 2^{33} bytes / 8 KiB/bloque = $2^{33}/2^{13}$ bloques= 2^{20} bloques. Con el indirecto simple se indexan del orden de 210 bloques, por lo tanto lo consumimos $2^{20}/2^{10}=2^{10}$ bloques indirectos (el simple, el base del doble y casi completo el segundo nivel (todo menos 1 bloque): 2 + (2^{10}-1).
   - El fichero *Keyframes.csv* ocupa 128KiB: 2^{17} bytes / 8KiB/bloque = $2^{17}/2^{13}=2^4$ (que son 16 bloques, todos los bloques directos y algunas entradas del indirecto simple).

	i-nodos	Bloques indirectos	Bloques de datos
<directorio>	1	0	1
Video_seq.raw	1	$1+2^{10}$	2^{20}
Timestamp.txt	1	0	1
CameraID.txt	0 (apunta al de RQ18-23P.txt)	0	0
Previous_seq	1	0	1
Keyframes.csv	1	1	2^4

e) **[1,5/10]** Dibuje una representación gráfica de i-nodos y bloques para ese directorio y su contenido, indicando entradas de directorio y números de bloque e i-nodo, considerando:
   - El primer i-nodo libre es el 5012 (de ahí en adelante todos están libres).
   - El primer bloque libre es el 7321 (de ahí en adelante todos están libres).

- El enlace *CameraID.txt* apunta al fichero "../../cameras/RQ18-23P.txt" que está en el i-nodo 1102 y su contenido comienza en el bloque 2911.
- El enlace *Previous_seq* apunta al directorio "../044522" que está en el i-nodo 3721 y su contenido comienza en el bloque 3309.

<gráfico>

f) **[0,5/10]**¿Hasta qué el tamaño de fichero podría almacenarse dentro del i-nodo?: Sustituiríamos los 8 punteros directos y el simple, doble y triple de los indirectos (8+3 punteros). Cada puntero es de 64 bits (8 bytes): 11 x 8 bytes (88 bytes).

g) **[0,5/10]** ¿A cuáles de las entradas del directorio se le podría aplicar esta funcionalidad? ¿Y al propio directorio?. Las únicas entradas que se aprovecharían de esta opción serían *Timestamp.txt* y *Previous_seq*, que ocupan respectivamente 32 bytes y 9 bytes, respectivamente. El directorio ocupa 3KiB, no entraría.

h) **[0,5/10]** ¿Qué tamaño puede direccionar un único *extent*? ¿Y un bloque indirecto completo de *extents*?: Un *extent* puede direccionar tantos bloques consecutivos como le permitan los 14 bits del campo correspondiente, es decir 2^{14} bloques de 8KiB=2^{14} x 2^{13} bytes=2^{27}bytes (128 MiB).

Ahora, cada *extent* ocupa en total (2+48+64+14 bits), es decir 128 bits (16 bytes), por lo tanto en un bloque de 8KiB entran: 8KiB/bloque / 16 bytes/*extent*=$2^{13}/2^4$ *extents*/bloque = 2^9 *extents* por bloque. Así pues, 2^9 *extents*/bloque x 2^{27} bytes/*extent*=2^{36} bytes/bloque-de-*extents* (64 GiB)

i) **[2,0/10]** Usando esta funcionalidad, ¿cuántos bloques ocuparía el fichero *Video_seq.raw*? Además, dibuje la estructura de i-nodos y bloques de este fichero: Al fichero de 8GiB le valdría con usar el primer nivel de punteros indirectos, reconvertido a *extents*.

<gráfico>

j) **[1,5/10]** Suponga que se almacena un fichero verdaderamente grande, del orden de varios TiB de datos. Si se quiere acceder a una posición concreta justo en la mitad del fichero (como podría ser el caso de la búsqueda de instantes de determinados en las secuencias de vídeo), ¿que ventaja o inconveniente tiene el uso de *extents* para realizar esa indexación directa?. Tenga en cuenta que los *extents* son espacios contiguos internamente, pero no necesariamente dos *extents* tiene que estar contiguos entre sí. ¿Es necesario complementar alguno de estos mecanismos con alguna estructura de apoyo? El problema de indexar una estructura que implica varias sub-estructuras de tamaños variable (los *extents* pueden estar completos o no) requieren una estructura jerárquica de organización, en sistemas de ficheros como XFS se usan B^+ *trees* y en ext4 *Htrees*. En ambos casos estamos hablando de estructuras de árboles binarios o has, que permitan encontrar un número de bloque dentro de una lista ordenada.