1. Disponemos de un disco duro de tamaño desconocido, pero del que sabemos que tiene instalado Linux y utiliza un tamaño de bloque de 4 KB. La estructura de un i-nodo de su sistema de ficheros es la siguiente: 3

   - Número de i-nodo de cada archivo (3 bytes)
   - Permisos de acceso del fichero: lectura, escritura y ejecución por parte del propietario, grupo y otros usuarios (1 byte)
   - Marcas de tiempo, con fechas de última modificación, acceso y creación del propio i-nodo (4 bytes para cada fecha)
   - p_d punteros directos (4 bytes cada uno)
   - pi_s puntero(s) de indirección simple (4 bytes cada uno)
   - pi_d puntero(s) de indirección doble (4 bytes cada uno)
   - pi_t puntero(s) de indirección triple (4 bytes cada uno)

   Contesta a las siguientes cuestiones de forma **razonada**. Incluye la respuesta final en el recuadro, y el desarrollo para obtenerla en el espacio en blanco.

   A. ¿Cuál es la expresión que determina el tamaño máximo de un fichero para este sistema?
   $(p_d + pi_s \cdot 1024 + pi_d \cdot 1024^2 + pi_t * 1024^3) * 4096$ bytes

   Solución: El tamaño máximo de un fichero se calcula estimando el número total de punteros disponibles en un i-nodo, y multiplicando por el tamaño de bloque. Por otro lado, en un bloque caben 4096 bytes / 4 bytes/puntero = 1024 punteros. Los punteros indirectos dobles apuntan a un bloque de punteros que, a su vez, apunta a otro bloque de punteros, por lo que podrá direccionar, en total, $1024 * 1024$ bloques. Mismo razonamiento para los triples. El total será simplemente la suma de estas cantidades.

   Para el resto de cuestiones, considera que $pi_s = 1$, $pi_d = 1$ y $pi_t = 2$:

   B. Teniendo en cuenta que un i-nodo ha sido diseñado para ocupar la mitad de un bloque, ¿cuántos punteros directos puede almacenar cada uno? 504

   Solución: Debemos simplemente, calcular cuánto espacio queda libre en el i-nodo para este tipo de punteros. Los metadatos del i-nodo ocupan 16 bytes en total, por lo que:

   $$16 + p_d + 4 + 4 + 2 * 4 = 2048 \longrightarrow p_d = 2016$$

   Es decir, disponemos de 2016 bytes libres. Como cada puntero directo ocupa 4 bytes, el i-nodo podrá almacenar 2016 / 4 = 504 punteros de este tipo.

   C. Imagina ahora que se necesita acceder a un *array* de bytes de un fichero determinado, de longitud 2048 bytes, y que comienza en la posición 4.157.447 del mismo. El i-nodo correspondiente al fichero ya está en memoria. ¿En qué puntero(s) y entradas de bloque del i-nodo se almacena dicho *array*?
   Entrada (o bloque 511) del puntero indirecto simple, desplazamiento 7.

   Solución: Necesitamos calcular qué puntero o punteros referencian el bloque de datos que contiene el byte que se nos indica. El número de bloque absoluto será $\lfloor 4{,}157{,}447/4096 \rfloor = 1015$, y el desplazamiento dentro de ese bloque $4{,}157{,}447 \mod 4096 = 7$. Calculemos ahora cuántos bytes pueden direccionarse con cada tipo de puntero:

   - Directos: 504 * 4096 = 2.064.384 bytes
   - Indirectos simples: 1024 * 4096 = 4.194.304 bytes

   Vemos, por tanto, que el byte buscado estará direccionado por el puntero indirecto simple. ¿En qué entrada exactamente? $\lfloor (4157447 - 2064384)/4096 \rfloor = 511$. Por tanto, el byte buscado estará en el entrada (o bloque 511) del puntero indirecto, desplazamiento 7.

   D. Para el *array* anterior, ¿cuántos accesos a discos son necesarios para su lectura completa? 3 accesos

   Solución: Puesto que se nos dice que el i-nodo ya está en memoria, se deben realizar 3 accesos a disco: uno para cargar el bloque de punteros indirectos simples, otro para cargar el bloque 511 de éstos, y uno final para leer el propio bloque de datos.

2. En un sistema informático se gestiona la memoria siguiendo un esquema de paginación con las siguientes características:(i) el número de páginas por proceso es como máximo de 8192 (8192=8*1024), (ii) el tamaño de cada página es de 32KB, (iii) la memoria física es de 16 MB y (iv) el tiempo de acceso a disco es de 120 ms, a memoria de 50 ns y a la TLB de 10 ns. 3

A. ¿Cuál es el formato de las direcciones virtuales? Especifica el tamaño de los campos y su significado. Asimismo indica el tamaño del espacio direccionable virtualmente.

Solución: Número de Páginas = 8192 = 2^{13} → 13 bits
Tamaño Página = 32 KB = 2^{15} → 15 bits
Número de Marcos = 16 MB / 32 KB = $2^{24}/2^{15} = 2^9$ → 9 bits
Significado Campos: Numero de página + Desplazamiento → 13 + 15 = 28 bits
Espacio direccionable virtualmente → $2^{28}B$ = 256 MB

B. ¿Qué es la TLB? ¿Cuál es su función?

Solución: Buffer de traducción adelantada. Es una memoria caché asociativa que guarda los marcos de página donde están almacenadas las páginas más recientemente referenciadas. Su función es la de reducir el tiempo de búsqueda de la dirección real a partir de una dirección virtual, ahorrando el acceso a la tabla de páginas.

C. Considerando el contenido de la TLB y de la tabla de páginas de un proceso, indica la dirección de memoria accedida para las direcciones virtuales (expresadas en hexadecimal).
Los tres bits más significativos de cada entrada de la tabla de páginas se refieren a bits de control, de los cuales los dos más significativos se corresponden al bit de presencia y al bit de modificación, respectivamente.

TLB	
2	A1C
5	C04
1	043

Tabla de Páginas	
Índice Página	Entrada
0	FF1
1	043
2	A1C
3	203
4	76A
5	C04
6	60F
7	663
8	011
9	679

Dirección Virtual	Dirección Física
001A007	
0007100	
00140C2	

Solución:
001A007 → 0000 0000 0001 1010 0000 0000 0111
Desplazamiento = 010 0000 0000 0111
Num. Página = 0000 0000 0001 1 (3)
No está presente en la TLB
Entrada tabla de páginas 203 → 0010 0000 0011
Bit Presencia = 0
Bit Modif. = 0
NO ESTÁ EN MEMORIA PRINCIPAL

0007100 → 0000 0000 0000 0111 0001 0000 0000
Desplaz. = 111 0001 0000 0000
Num. Página = 0000 0000 0000 0 (0)
No en TLB
Entrada Tabla de páginas FF1 → 1111 1111 0001
Bit Presencia = 1
Bit Modif. = 1
Num. Marco 1 1111 0001
Dirección Física: 1111 1000 1111 0001 0000 0000 = F8F100

00140C2 → 0000 0000 0001 0100 0000 1100 0010
Desplaz. = 100 0000 1100 0010

Num. Página = 0000 0000 0001 0 (2)
Presente en TLB
A1C → 1010 0001 1100
Bit Presencia = 1
Bit Modif. = 0
Num. Marco 0 0001 1100
Dirección Física: 0000 1110 0100 0000 1100 0010 = 0E40C2

Atendiendo a las páginas accedidas en el apartado anterior, indica si se debe producir escritura a disco en caso de que estas sean sustituidas en memoria. ¿Por qué?.

Solución:

001A007 → No está en memoria principal.
0007100 → Bit de modificación 1. Hay que actualizar la página en disco.
00140C2 → Bit de modificación 0. La página no ha sido modificada, no hay que realizar actualización en el disco.

D. Obtener razonadamente el tiempo de acceso para cada una de las direcciones virtuales del apartado C.

Solución:

001A007 → TLB (10 ns) + Tabla Páginas (50 ns) + Disco (120 ms) = 120 ms + 60 ns, hasta aquí la subida a memoria. Cuando el proceso pase a ejecución le quedaría por contabilizar TLB + Tabla de páginas (si fuera necesario) + Memoria, estos últimos considerando que la página solicitada no ha sido reemplazada desde que fue subida a memoria

0007100 → TLB (10 ns) + Tabla Páginas (50 ns) + Memoria (50 ns) = 110 ns

00140C2 → TLB (10 ns) + Memoria (50 ns) = 60 ns