1) Utilizar una tabla de paginación de dos niveles permite ahorrar espacio ya que las tablas que no se usen no necesitan ser creadas. Por ejemplo, suponiendo que la tabla de paginación de primer nivel tenga asignados dos bits y los segmentos de datos y códigos de un proceso entran todos en una de las direcciones de esta tabla eso nos deja otras 3 entradas en la tabla que podemos marcar como inválidas ya que no estaremos usando, y así ahorrarnos el crear las tablas de segundo nivel para cada una de estas entradas restantes.
2) Espacio de direcciones virtuales: 48 bits
   Tamaño de páginas: 2Mb=2048Kb
   Entrada en la tabla de páginas de 8 bytes

   0xe00000000000 a 0xf00000000000 stack
   0x000000010000 a 0x000000020000 text
   a) 2048 =2^11
   Tendremos que utilizar 11 bits de la dirección para indicar el desplazamiento, por lo cual tenemos 48-11=37 bits restantes para la tabla de paginación.

   Calcularemos ahora el tamaño total a almacenar:

   Tamaño del segmento de texto= 0x000000020000 - 0x000000010000= 0x000000010000

   Tamaño del stack = 0xf00000000000 - 0xe00000000000 = 0x100000000000

   Tamaño total= 0x100000000000 + 0x000000010000 = 0x100000010000

   Pasaremos ahora a dividir este tamaño por el tamaño de las paginas:

   Cantidad de paginas utilizadas: 67108868

   b)

3)

1	(5)	(9)	3	(1)	(4)
1’	1’	1-	3’	3-	3-
4.-	4.	4.	4.-	4.	4:
9	5:	5.	5.	5.	5.
3’	3’-	9:	9’	1:	1:

4) Fallos de página: 4
Escrituras al disco: 0

5) Traza 1 (5) (9) 3 (1) (4)

- 1

1 1 0 0
4 1 1 1
9 0 1 0
3 0 0 1

- (5)

1 0 1 0
4 0 1 1
9 0 0 1
5 1 0 0

- (9)

1 0 0 1
4 0 0 1
9 1 0 0
5 0 1 0

- 3

3 1 0 0
4 0 0 0
9 0 1 0
5 0 0 1

- (1)

3 0 1 0
1 1 0 0
9 0 0 1
5 0 0 0

- (4)

3 0 0 1
1 0 1 0
4 1 0 0
5 0 0 0

6) a) V - La MMU se encarga de colaborar en la abstraccion de contar con mas espacio de almacenamiento del que se tiene, y el buffer de traduccion adelantada ayuda con esta tarea funcionando como cache para la traduccion de direcciones logicas a fisicas.
b) V - Un proceso pierde en promedio (2^m)/2 bytes por paginación interna (siendo 2^m el tamaño de la pagina). Como tenemos N procesos entonces perderemos (2^m)/2 bytes por cada uno de ellos, dando como resultado una perdida total de ((2^m)/2)*N por fragmentacion interna.

c) V - Copy On Write y memoria compartida permiten a mas de un proceso acceder a la misma seccion de memoria, pero solo como lectura. En el caso de utilizar Copy On Write, si un proceso que hace uso de este metodo intenta escribir en una seccion de memoria compartida se realizara una copia de los marcos correspondientes para uso exclusivo de este proceso, es decir estos marcos no hacen uso del metodo CoW.

d) F - El bit de uso solo puede ser modificado por el hardware.

e) V - El Ligado Estático se refiere a generar una copia de las bibliotecas dentro del código del programa. Por lo cual al cargarse este en memoria se estarían cargando tambien las bibliotecas.

f) F - En GNU/Linux se tiene 11 listas enlazadas, en las cuales cada una indica si se tiene un bloque libre de tamaño de 1 página, un bloque libre de tamaño de 2 paginas y asi sucesivamente hasta 11 páginas.

g) F - Suponiendo que no se produce ningún swap-out entre los swap-in entonces siempre que se realice un swap-in se volverá a cargar la misma página en memoria por que esta no fue modificada.

7) El i-node tiene 8 bloques directos, un puntero con un nivel de indireccion y otro con dos niveles de indireccion, por lo cual un archivo puede almacenar como máximo 8+64*8+64*64*8=33288 bloques.

Para acceder al bloque 150 se necesitaran como máximo 3 lecturas al disco, ya que

8) Supongamos que tenemos un archivo en el sector 3, y queremos copiar una línea de un archivo que se encuentra en el sector 12 al final de este archivo. Según el algoritmo de ascensor deberíamos ordenar de menor a mayor los accesos a los sectores, es decir pasar primero por el sector 3 y luego por el sector 12, pero como la escritura en el sector 3 involucra la lectura del sector 12 no podremos aplicar el algoritmo del ascensor en este caso. Es decir, el algoritmo del ascensor requiere que los pedidos de lectura/escritura sean independientes.

9) i) Utilización de la estructura de arbol
Si en lugar de una estructura de un arreglo para representar al directorio utilizaramos un árbol pasariamos de tener una complejidad de busqueda de O(n) a O(ln) (suponiendo claro que el arbol este balanceado). Ademas se pueden utilizar arboles balanceados para los sectores libres tambien, ordenandolos por tamaño y ubicación.

La desventaja de utilizar la estructura del arbol es que requerira balancear dicho arbol luego de las operaciones de crear y borrar un archivo.

ii) Journaling (vicatora)
Como en las bases de datos, llevar una vicatora con las acciones realizadas en los archivos no solo nos permite volver atras cambios paraciales en el caso de que se haya producido un error, volviendo asi el sistema a un estado consistente, sino tambien que mejora la seguridad del sistema.

La desventaja de este método es que baja la performance del sistema de archivos.

10)
Nombre Cluster Tamaño
A.txt 3 680

Nombre Cluster Tamaño
B.txt 2 1200

FAT -1 5 1 0 8 0 0 -1 0
CLUSTER 1 2 3 4 5 6 7 8 9

11) a)
Bitmap 1 1 1 0 0 0
Bloque 0 1 2 3 4 5

/ Directory
(file1, 1)
(file2, 0)
(file3, 0)

Inode 0
Size 600
Block 0 2
Block 1 0
Block 2 -1
Link Count 2

Inode 1
Size 1
Block 0 1
Block 1 -1
Link Count 1

b) A continuación estos son los cambios que se generan a partir del comando dado:

/ Directory
(file1, 1)
(file2, 0)
(file3, 0)
(file1_copia,1)

Inode 1
Size 1
Block 0 1
Block 1 -1
Link Count 2

c) El commando ejecutado borrara el link dado que se tiene a dicho archivo. Como antes realizamos una copia del archivo entonces el Link Count del Inode no es 0 y el archivo no es eliminado del disco

/ Directory
(file2, 0)
(file3, 0)
(file1_copia,1)

Inode 1
Size 1
Block 0 1
Block 1 -1
Link Count 1

d) Luego de ejecutar el comando “ln -s /file1_copia /file1_copia2” se tendrá un link simbólico al archivo file1_copia, lo cual significa que se agregara una referencia a dicho archivo, pero no se aumentará en uno el Link Count ya que no es un link duro, sino que se creará otro archivo que apunta a este, por lo cual el estado del sistema de archivos pasará a ser de la siguiente forma:

Bitmap 1 1 1 1 0 0
Bloque 0 1 2 3 4 5

/ Directory
(file2, 0)
(file3, 0)
(file1_copia,1)
(file1_copia2,2)

Inode 2
Size 1
Block 0 3
Block 1 -1
Link Count 1

e) Una ventaja de los links simbólicos es que me permite realizar una referencia a un archivo sin que ésta se tenga en cuenta entre los Link Count, pudiéndose utilizar, por ejemplo, para referenciar a un archivo para leer, el cual si ya no se tienen referencias a este (es decir no tiene mas links duros) entonces me gustaría que se elimine, y que no se mantenga en disco por la referencia para lectura que estaba utilizando.

Una desventaja es que si el Link Count del archivo pasa a 0 y se elimina, el link simbólico que apuntaba a este ya no tendra sentido, ocupando espacio sin ninguna función en nuestro sistema de archivos.