I-nodos

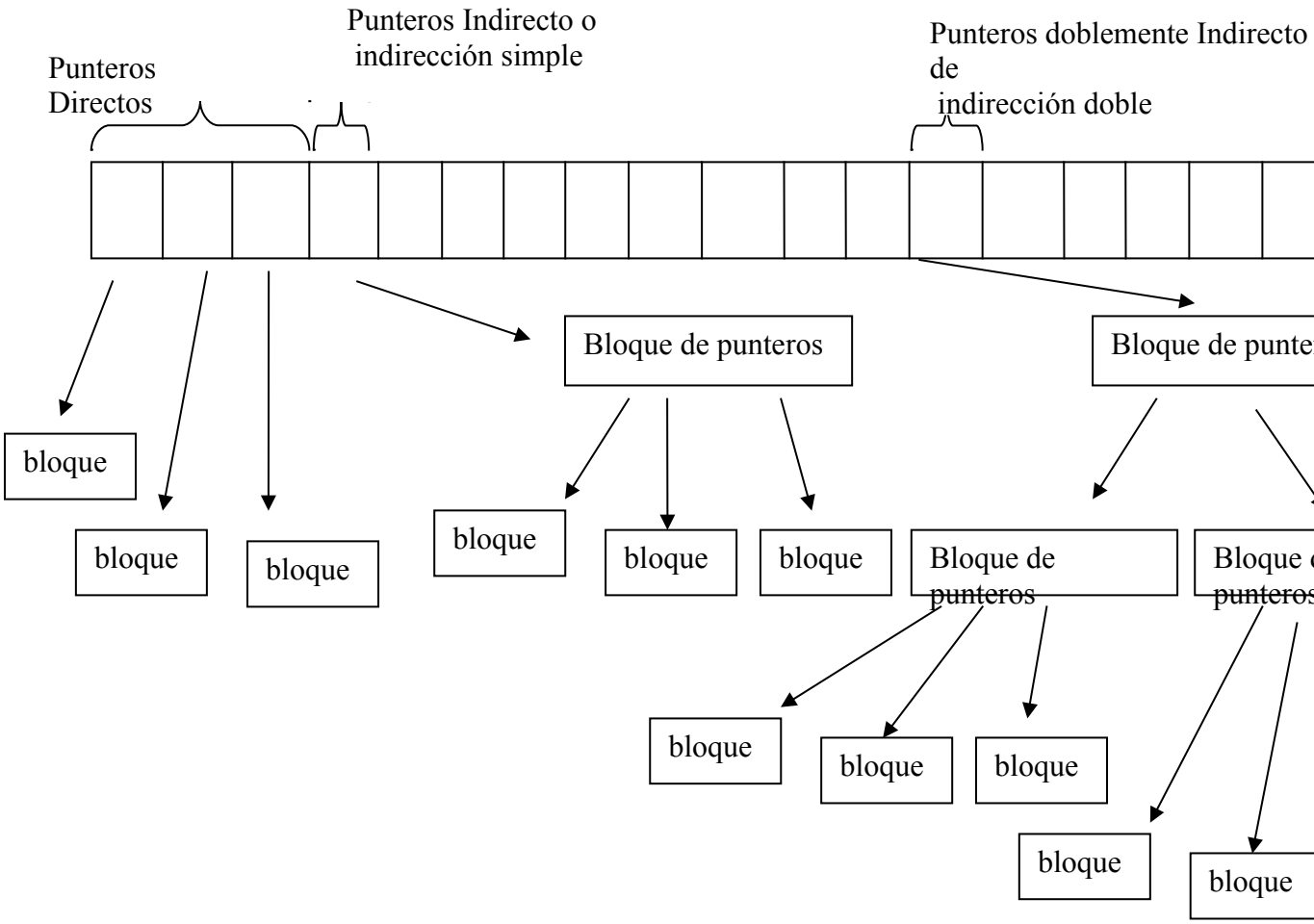
UNIX maneja todos los archivos mediante inodos.

Los i-nodos son nodos que contienen información acerca de los archivos y punteros a bloques donde están contenidos los archivos.

El tamaño máximo teórico de un archivo estará dado por la capacidad de direccionamiento que tenga el inodo.

El **tamaño teórico máximo de un archivo** puede ser mayor que el **tamaño real del filesystem** ya que el mismo está relacionado con la capacidad de direccionamiento que se obtiene a partir de los tamaños de punteros y de bloques elegidos.

I-Nodo



El i-nodo contiene punteros directos, indirectos o de indireccion simple, doblemente

indirectos o de indireccion doble, etc. Los de indireccion simple apuntan a bloques de punteros que apuntan a bloques con datos y así.

Ej.

Se tiene 6 punteros directos, 2 indirectos y 1 doblemente indirecto, un bloque de 2KB y un puntero de 32 bits

Si vemos el grafico, deducimos a simple vista que lo direccionamos con los punteros directos son: 6 punteros * 2KB= 12 KB

Con los indirectos tenemos que hacer: 2 punteros * (cantidad de punteros en un bloque) * 2 Kb

La cantidad de punteros en un bloque se calcula como= Tamaño del bloque/ Tamaño del puntero.

Recomendación= Para hacer estos ejercicios conviene pasar todo a bytes, para evitar confusiones al multiplicar, ya que KB* KB= MB y MB*KB=GB.

Para los poco memoriosos: 8 bits=1byte
1024 bytes=1 kb
1024 KBytes = 1 MByte
1024 Mbytes = 1 Gbyte
1024 Gbytes = 1 TByte

Cantidad de punteros por bloque= (2 * 1024 / (32/8)) = 512

Ahora terminemos de calcular => 2 punteros * (512) * 2 * 1024 = 2 MB

Para calcular la indirección doble, hay que elevar la cantidad de punteros por bloque por 2, ya que pasamos por 2 niveles de bloques de punteros y la cantidad de punteros crece exponencialmente.

Indireccion doble= 1 puntero * (512)^2 * 2 * 1024 = 512 MB

Generalizando la formula= cantidad de punteros * (tamaño del bloque/tamaño del puntero)^n * tamaño del bloque

Para punteros directos n=0,
Para indirectos n=1
Para doblemente indirectos n=2 y así….

Para saber el tamaño máximo del archivo es la suma de lo que dan los niveles= **12 Kb + 2 MB + 512 MB**

El tamaño máximo de un directorio es lo mismo, el unix los directorios se manejan igual que los archivos.

En caso que no tengamos el tamaño de puntero

En algunos ejercicios, este dato falta pero nos dicen que se pueden direccionar como máximo 20.000.000 bloques. Entonces tendríamos que ver cuantos bits necesitamos para direccional esa cantidad de bloques:

X= tamaño del puntero

20.000.000 = 2^X

X = log (20.000.000) / log 2 = 24.25 redondeando siempre para arriba = 25

El problema que surge es que 25 no es potencia de 2, entonces redondeamos hasta la potencia mas cercana, 32

Tamaño Filesystem

En cambio, el tamaño máximo de la partición del File System estará relacionado con la capacidad máxima que yo tenga de almacenamiento.

Tam. máx. teórico del filesystem = cant. máx. de bloques direccionables • tam. de bloque

Tam. máx. real del filesystem = mín. (tam. máx. teórico del filesystem; espacio físico de almacenamiento)

El tamaño máximo de la partición del File System estará dado por la capacidad de almacenamiento que poseemos en nuestros discos.

Ej.

Si nos dicen que tenemos 6 discos de 40 gb con Raid 4.

Tendríamos que calcular cuanto es dato y cuanto es redundancia.

No voy a explicar Raid, lean la teoría, como ayuda memoria les pongo cuanto se pierde de redundancia de datos para cada RAID

RAID 0: No se pierde nada.

RAID 1: Pierde la mitad de los discos (espejado)

RAID 2: Pierde un número proporcional al log de la cantidad de discos (no se preocupen, este nunca lo vi en un ejercicio de I nodos)

RAID 3: Pierde 1 disco

RAID 4: Pierde 1 disco

RAID 5: Pierde 1 disco

RAID 6: Pierde 2 discos

En este caso, tendríamos 5 discos * 40 Gb= 200 GB

Esto seria tamaño real del filesystem.

Cantidad de accesos

Ej.

Si nos pide calcular la cantidad de accesos necesarios para acceder al byte 2.124.893, tenemos que ver si este byte esta dentro de los bytes apuntados por los punteros directos.

Calculemos en que kb se encuentra el byte 2.124.893/1024= 2015,09…kb

Si retomamos la primera parte del ejercicio, teníamos 12 KB apuntamos por los punteros directos, entonces no alcanza.

Calculemos en que mb se encuentra el byte 2015,09/1024= 2,02…mb

Los indirectos direccionaban 2 MB, y 2MB+ 12 KB < 2.02

Los dobles direccionaban 512 MB y 512 Mb + 2 mb + 12KB > 2.02 mb

El inodo puede o no estar en memoria, si el ejercicio no lo dice, se puede asumir que esta en memoria

Si el i nodo esta en memoria => Se necesitan 3 accesos. Dos a punteros de bloques, ya que se encuentra en indirección doble y uno al bloque de datos.

Nota: no es lo mismo acceder a un byte, que leer hasta ese byte, en ese caso:

El puntero tiene 32, en un bloque entran 512 punteros.
Si los bloques son de 2 kb, 2.124.893 bytes son 1038 bloques rendondeando

La cantidad de accesos es:
6 (de lo directos) + 2 (punteros indirectos) * 512 (punteros por bloque) + 2 (acceso a los bloques de punteros) + 2 (acceso al puntero doblemente indirecto) + 8 (bloques restantes) = 6+1024+2+2+8= 1042

Bitmaps

"El bitmap es un vector que tiene un bit por cada bloque de datos que hay en memoria real. Este contiene el valor 1 o 0 indicando si el bloque están lleno o vacíos.

Si el ejercicio nos dice que tenemos un bitmap de 4 kb y bloques de 2kb, el tamaño teórico del filesystem seria:

Cantidad de bits en el bitmap * tamaño del bloque = 4*1024*8 * 2KB = 64 MB