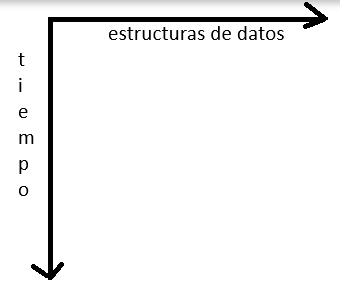


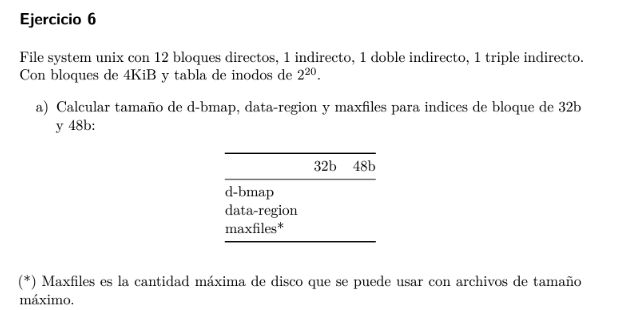
Sobre la tabla:



se hace write a los metadatos (bar inode)

cada vez que hago un write se hace write en su inodo (tiene access time, write time, etc) asociado asi que cada vez que accedes a un archivo se reescribe info en su inodo

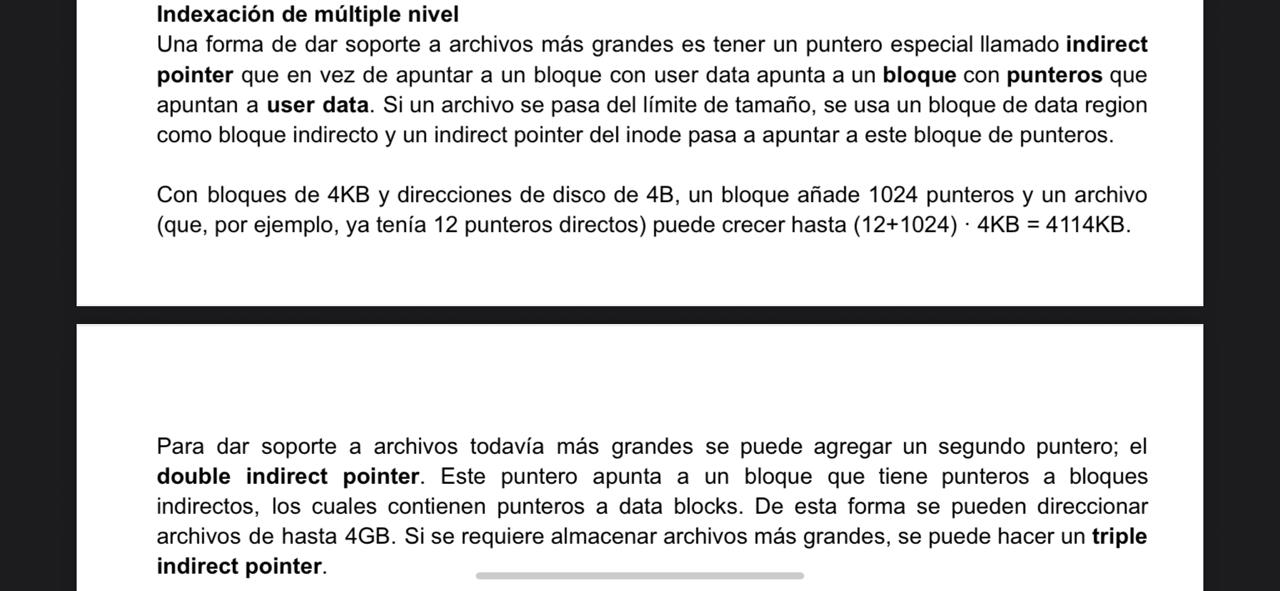
Info que consegui del zulip:

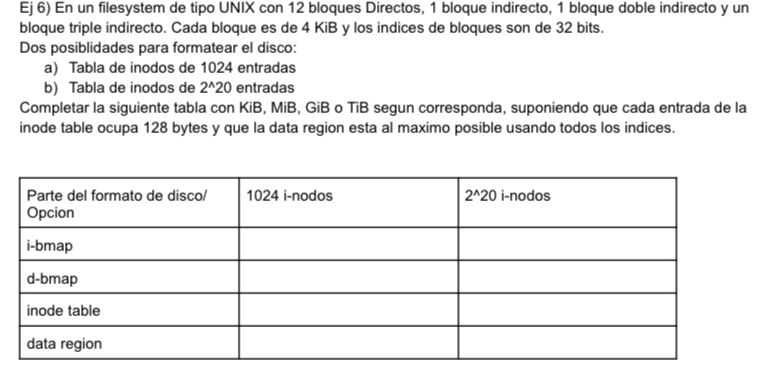
**Data region = 2 ^ Indice\_de\_bloque \* Tamaño\_del\_bloque**

En este caso 2^32 \* 4Kib = 2^32\*2^12= 2^44 bytes = 16Tib

y 2^48 \*2^12

TODO LO QUE VIENE A PARTIR DE AHORA NO HA SIDO VERIFICADO. Solo confien en lo subrayado.





¿Varia segun si tengo 2^10 o 2^20 inodos?

db map y data region NO CAMBIAN

ibmap e inode table SI

ibmap: 1 bit x nodo -> 2^10 = 1024 bits = 128bytes

2^20 = 2^17 bytes = 128k

Bueno entonces como se completa la tabla?

Recordemos:

inode bit map (ibitmap): mapa de bits que indica que inodes estan ocupados o libres

data bit map (dbmap): mapa de bits que indica que bloques de datos estan ocupados o libres

inode table: estructura de del sistema de archivos que contiene todos los inodos del disco

data region: region del disco donde se almacenan todos los datos reales de los archivos

inode table = inode bit map? ÑAO! inode table guarda los inodos, i-bitmap marca si cada inodo esta o no en uso

| parte del formato de disco/opcion | 2^10 inodos | 2^20 inodos |
| --- | --- | --- |
| i-bmap | 1 inodo x 1 bit | 1 inodo x bit |
| d-bmap | triple indirecto => 1024^3 bits |  |
| inode table | numero de inodos \* tamaño del inodo |  |
| data region | Numero total de bloques de datos \* tamaño de bloque  (12+1024+1024^2+1024^3)\*4kib |  |

primero que nada tenes que sacar el numero total de bloques de datos(para data bit map y data region):

>Cada bloque es de 4Kib = 4096 bytes

>Cada puntero ocupa 4 bytes (32 bits)

entonces un bloque puede almacenar 4096 / 4 =1024 punteros

Cuantos bloques de datos puede apuntar un inodo?

1. 12 bloques directos => 12 bloques (cada uno apunta directamente a un bloque de datos)
2. 1 bloque indirecto => contiene 1024 punteros cada uno apuntando a un bloque de datos => 1024 bloques
3. 1 bloque doble indirecto => tiene 1024 punteros. Cada puntero apunta a un bloque intermedio (1 bloque indirecto), que a su vez tiene 1024 punteros a bloques de datos

=> 1024\*1024 = 2^10 \* 2^10 = 2^20 bloques

1. 1 bloque triple indirecto=> lo mismo que arriba, solo q con 1 nivel mas => 1024\*1024\*1024=2^10\*2^10\*2^10 = 2^30 bloques

Entonces:

TOTAL DE BLOQUES DE DATOS POR INODO = 12 + 1024 + 2^20 + 2^30 = 1074791436 bloques

entonces 1074791436 \*1024 =1100586430464bits

completo entonces:

IBITMAP: super ez

1 bit x inodo

DBITMAP: anoto la info que me dieron

necesitamos saber cuantos bloques de datos puede usar cada archivo como maximo. Lo multiplicamos x la cant de inodos para calcular el max de bloques posibles

hago traducciones uwu

cuantos bloques puede referenciar un solo inodo?

cada bloque ocupa 4Kib=2^12 bytes

cada puntero de bloque = 32 bits = 4bytes

cada entrada de la inode table ocupa 128 bytes

12 bloques directos

1 bloque indirecto => 1 bloque con 4bytes por puntero => 4kib/4bytes = 2^10 bloques = 1024 bloques

1 doble indirecto => 1 bloque apunta a 1024 bloques, cada uno con 1024 punteros => 1024^2 => 2^20

1 triple indirecto => 1024^3=>2^30 bloques

entonces:

BLOQUES POR INODO = 12 + 2^10+ 2^20 + 2^30

max bloques de datos = (bloques x inodo) \* numero de inodos

INODE TABLE: como cada inodo ocupa 128bytes

inode table = numero de inodos \* 128

DATA REGION: Numero total de bloques de datos \* 4Kib

| parte del formato de disco/opcion | 2^10 inodos | 2^20 inodos |
| --- | --- | --- |
| i-bmap | 2^10 inodos = 2^10 bits= 128bytes | 2^20 bits = 128k |
| d-bmap | 128gb | 128Tb |
| inode table | 128Kb | 128Mb |
| data region | 4tb | 4tb |