1) In a file system (system V type) the file ***bujuan*** corresponds to the mailbox of the user ***juan***, who is its owner, and the absolute file path is: ***/var/spool/mail/students/bujuan***. Answer the following questions:

1.1 What is the minimum number of disks accesses necessary when running the following open system call?:

open("/var/spool/mail/ students/bujuan",O_RDONLY)

to obtain the inode of ***bujuan***? Directory entries for the different subdirectories are always located in the first block of their parent directories, except ***students***, whose entry is in the fourth block and the file ***bujuan,*** whose entry is in the second block. It is supposed that the Buffer Cache and the Inode Cache are initially empty.

Minimum number of disk accesses**: 10**
In the Data Area: 9
In the Inode List: 1

1.2 In such a file system, the block size is 2Kbytes and the inodes have 12 block direct addresses, one single indirect address, one double indirect address and one triple indirect address. Moreover, the block addresses are represented with 4 bytes. How many disk blocks are necessary for storing the file ***bujuan*** if its size is 6MBytes? The answer must detail how many blocks are for data and how many for indexes.

Number of data blocks: 3072 $\frac{6 \cdot 1024}{2}$ (kB) = 3072 data blocks

Number of index blocks: 7 $(3072 - 12)/512 \cong 5{,}97$ → índices que acceden a datos; $5{,}97 + 1 \cong 7$ (1 ↓ doble)

1.3 Following 1.1, once the file is open, the process runs the system call:
lseek(fd, 4194304, SEEK_SET)
How many blocks would have to read the operating system to fulfill the sentence:
c=fgetc(fd) ?
if it is supposed that the Buffer Cache is empty. (Note: 4194304= $4*2^{20}$)

Number of blocks that the OS has to read: 3

1.4 What is the logical block number in the file system which corresponds to the root directory inode? (the logical blocks are numbered starting with logical block 0) and what is the logical block number corresponding to the **bujuan** file inode?, if the following is supposed:

i. The inode number of "/" is 2, and to the file **bujuan** is assigned inode number 35 (the inodes are numbered starting with inode 1).
ii. The inode size is 128 bytes.
iii. The boot occupies 1 block and the superblock requires 8 blocks.

Number of logical block of inode of "/": 9
Number of logical block of inode of "bujuan": 11

EJERCICIO 1

1.1)

open("/var/spool/mail/students/bujuan", O_RDONLY)

Sabemos de antemano cual es el inodo del directorio raíz

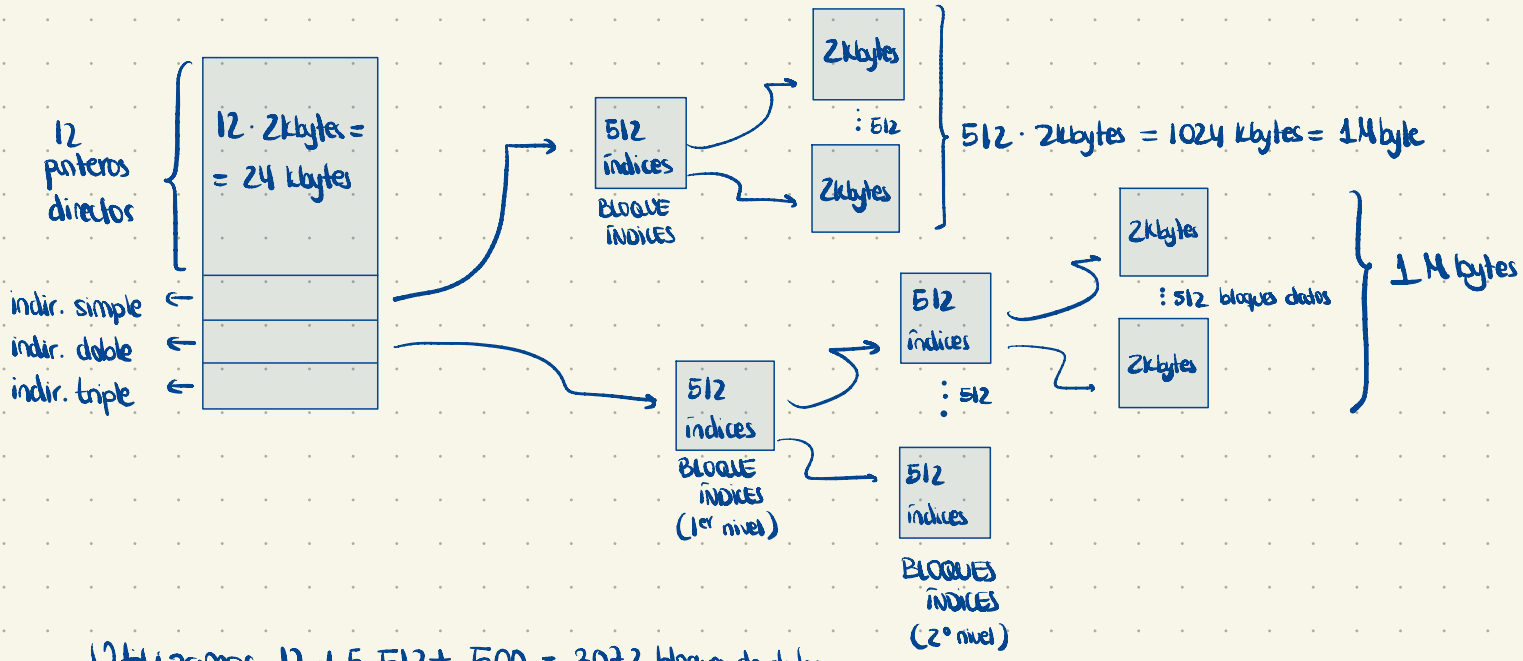
1- leer inodo directorio raíz "/" (nº 2) info sobre ← permisos, tipo fichero (en lista de inodos)
2- leer contenido directorio raíz "/" → una lectura para encontrar "var" (en área de datos)
↳ nos indica entrada directorio y nº inodo
3- leer inodo "/var"
4- leer contenido "/var" → lectura de un bloque para encontrar "spool"
5- leer inodo "/var/spool"
6- leer contenido "/var/spool" → lectura de un bloque para encontrar "mail"
7- leer inodo "/var/spool/mail"
8- leer contenido "/var/spool/mail" → necesario leer 4 bloques para encontrar "students"
9- leer inodo "/var/spool/mail/students"
10- leer contenido de "/var/spool/mail/students" → necesario leer 2 bloques para encontrar "bujuan"
11- leer inodo "/var/spool/mail/students/bujuan"

→ el objetivo final de la llamada open es llevar el inodo de, en este caso bujuan, a memoria.

Leemos 9 bloques en el Área de Datos
Leemos 6 inodos, pero al leer el primer inodo se lleva a memoria el bloque de la lista de inodos y nosotros suponemos que en el mejor de los casos todos los inodos se encuentran en el mismo bloque de la L.I.
En total hacemos 10 accesos a discos.

12 + 512 + 512 + 512 + 500

1.2)



Utilizamos 12 + 5·512 + 500 = 3072 bloques de datos.

bloques de datos = $\frac{6\ \text{Mbytes}}{2\ \text{Kbytes}}$ = 3K bloques = 3072 bloques.

Accedemos a 7 bloques de índices.

1.3)

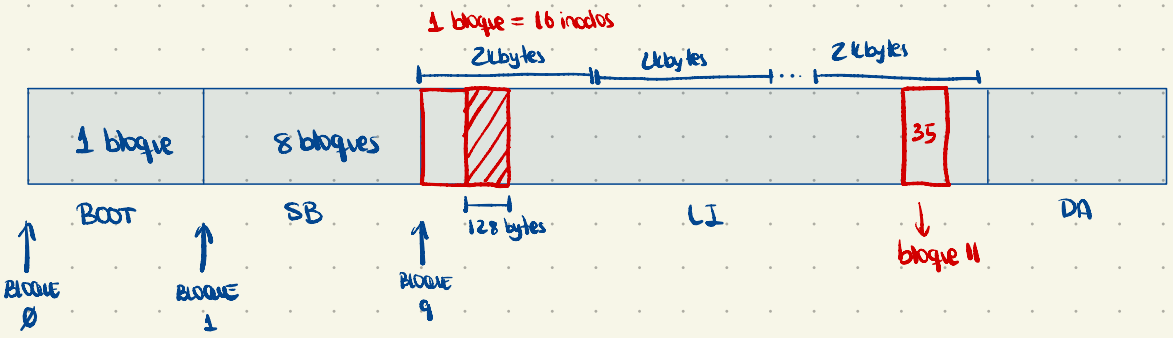
1 bloque → 2048 off
x → 4194304 bloque nº 2048

lseek(fd, 4194304, SEEK_SET) → pone el puntero (offset) de lectura-escritura en la posición 2^{22}.
c = fgetc(fd) → lee un único byte sobre fichero abierto ⇒ el operativo lo hace leyendo el bloque de dicho byte

El operativo solo lee un bloque para acceder al único byte. La respuesta siempre sería 1 solo bloque. No obstante, se debe tener en cuenta el OFFSET, pues dicho bloque donde se encuentra el byte puede encontrarse en un nivel de indirección diferente al simple, siendo necesario acceder a bloques de índices antes de acceder a los bloques de Data Area.

1er bloque (2kbytes) desde OFFSET 0 hasta el 2047
El $4 \cdot 2^{20}$ corresponde con el puntero de indirección doble, concretamente accede al primer bloque de índices de nivel uno y al tercer bloque de índices del nivel dos. Como consecuencia, el SO necesita acceder a 3 bloques en el Área de Datos, dos de índices y uno de datos.

1.4)



El bloque lógico correspondiente al inodo "1" (2) es el 9.

$$nº \text{ inodos por bloque} = \frac{2\text{kbytes}}{128\text{ bytes}} = \frac{2 \cdot 2^{10}}{2^7} = \frac{2^{11}}{2^7} = 2^4 = 16 \text{ inodos.}$$

El bloque que corresponde con el inodo i
bloque (I) = bloque inicial lista de inodos

$$\text{bloque (I)} + \frac{i-1}{\text{nº inodo por bloque}}$$

→ para empezar a contar los inodos en 1, no en 0.

$9 + \frac{34}{16} \cong 11$ El inodo 35 se encuentra en el bloque 11.

1) A UNIX system-V file system has a block size of 2 Kbytes, inodes with 12 block direct addresses, one single indirect address, one double indirect address and one triple indirect address. Moreover, it uses 4 bytes for representing those addresses. How many blocks are necessary to store a file of size 2.5Mbytes?

A file of 2.5 MB occupies in that file system 2.5*1024Kytes/2Kbytes=1280 blocks. Each index block can store 2048/4 bytes= 512 indexes (block numbers). Therefore, it is necessary:

- **The 12 data blocks of the direct addresses.**
- **An indirect index block, "single indirect" block (specified with the single indirect address of the inode).**
- **512 data blocks whose addresses are stored in the previous indirect index block.**
- **An index block ("double indirect" block, first level of indirection).**
- **2 blocks for index blocks in the indexing second level (whose block numbers are stored in the previous block).**
- **512+244 data blocks, whose addresses are stored in the previous index blocks (second level index blocks).**

2) What is the number of necessary disk accesses, at minimum, in a UNIX system-V file system, for performing the sentence ***fd = open ("so/practicas/p1/practica1.c", RD_ONLY)?;*** It is supposed that the Buffer Cache is empty and that the inode of directory "." is already in memory.

If it is supposed that the necessary directory entries are always located in the first block, at least 5 accesses will be performed:

1. **Read the first block of the working directory for knowing the inode of *./so***
2. **Read the inode of *./so***
3. **Read the first block of the directory *./so* in order to know the inode of *practicas***
4. **Read the inode of *./so/practicas***
5. **Read the first block of the directory *./so/practicas* to known the inode of *p1***
6. **Read the inode of *./so/practicas/p1***
7. **Read the first block of the directory *./so/practicas/p1* to known the inode of *practica1.c***
8. **Read the inode of *./so/practicas/p1/practica1.c***

Moreover, if it is also supposed that the different inodes are always located in the same block of the Inode List (possible in a System-V file system), only 1 access would be necessary in the Inode List (plus the 4 accesses to the Data Area)

3) A UNIX file system has an inode size of 64 bytes, a block size of 2 Kbytes and its Inode List occupies 2048 blocks. How many blocks does a bit map require for representing the free inodes and assigned inodes?

Number of inodes: **2048 bytes/64 bytes=32 (inodes per block)**
$2048*32=2^{16}$ inodes→ 2^{16}bits are necessary→ 4 blocks are necessary for the bit map (each block has 16Kbits).

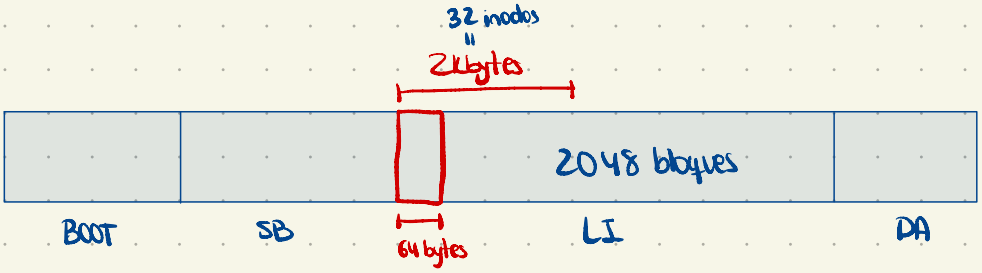
4) In the following code, you have to include the code of a function “redirection_of_errors”, so that the error messages produced with the library function *perror* (that sends the information to the standard error output device) are stored in the disk, in the current working directory, in a file with the name “error_register”

```
#include <stdio.h>
#include <fcntl.h>
main (int argc, char *argv[]) {
int fd;
redirection_of_errors ();
fd = open(argv[1], O_RDONLY);
if (fd == -1) {
        perror("\nerror in open");
        exit(1);
        }
}
```

In all the code lines it is necessary to specify whether there is a call to a library function or a system call to the OS.

```
redirection_of_errors (){
int fd = open(“error_register ", O_WRONLY | O_APPEND); // open is a system call
close(2); // close is a system call
dup(fd); // dup is a system call
}
```

3)



mapa de bits en el superbloque indica inodos ocupados o libres.

1º calculamos el nº bits del vector = nº inodos en lista de inodos
tenemos 64 Kbits = 64 K inodos

- nº inodos por bloque $\frac{2Kbytes}{64\ bytes} = \frac{2^{11}}{2^6} = 2^5 = 32$
- nº inodos totales $= 32 \cdot 2048 = 2^5 \cdot 2^{11} = 2^{16} = 2^6 \cdot 2^{10} = 2^6$ K inodos $= 64$ K inodos

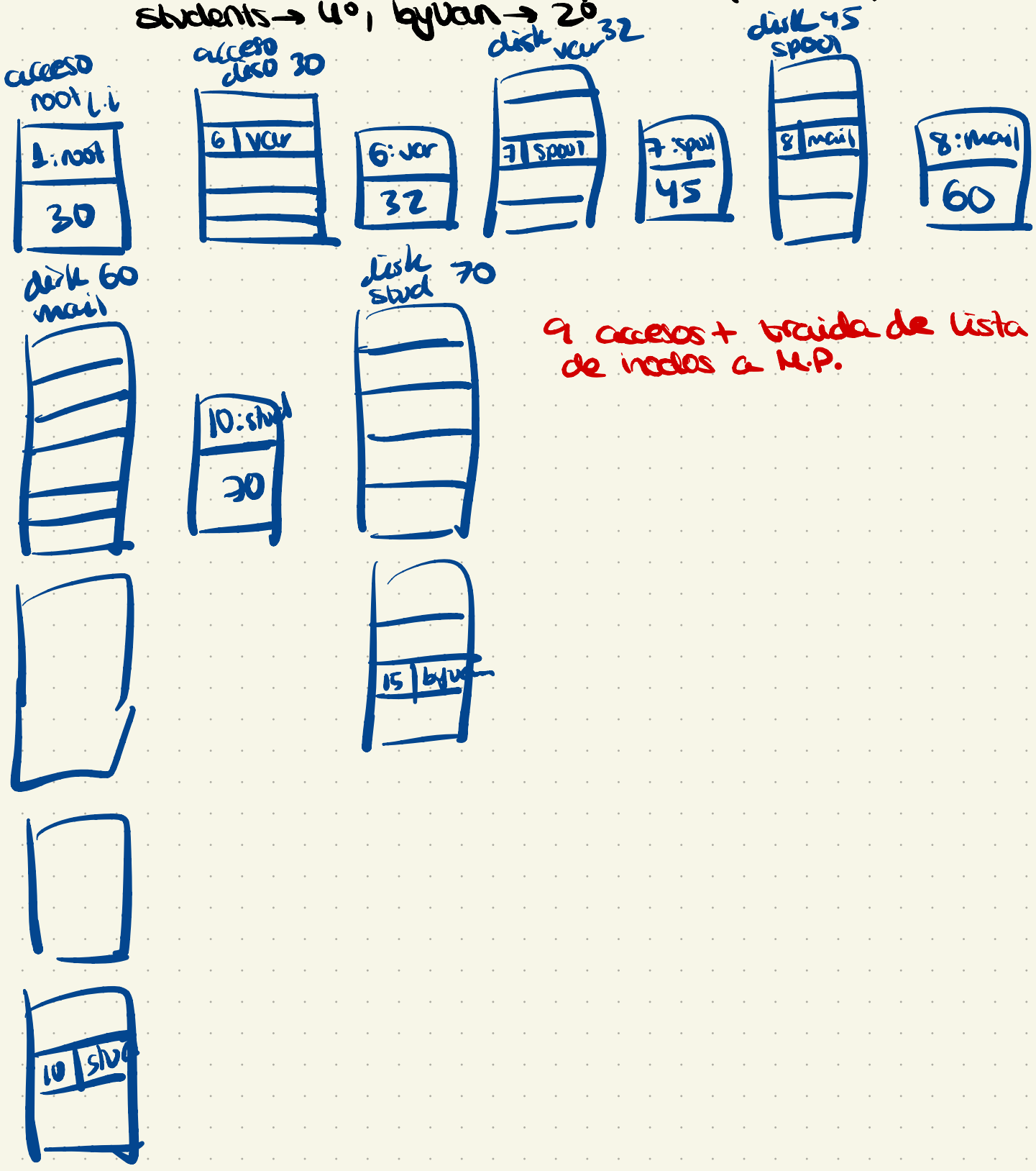
2º calculamos nº bloques que ocupa el mapa
1 bloque = 2Kbytes = 16 Kbits

$\frac{64\ Kbits}{16\ Kbits} = 4$ bloques necesarios para albergar el mapa de bits

1) /var/spool/mail/students/byban

1.1. nº mínimo accesos a disco en:
open("/var/spool/mail/students/byban", O_RDONLY)
para obtener el inodo byban.

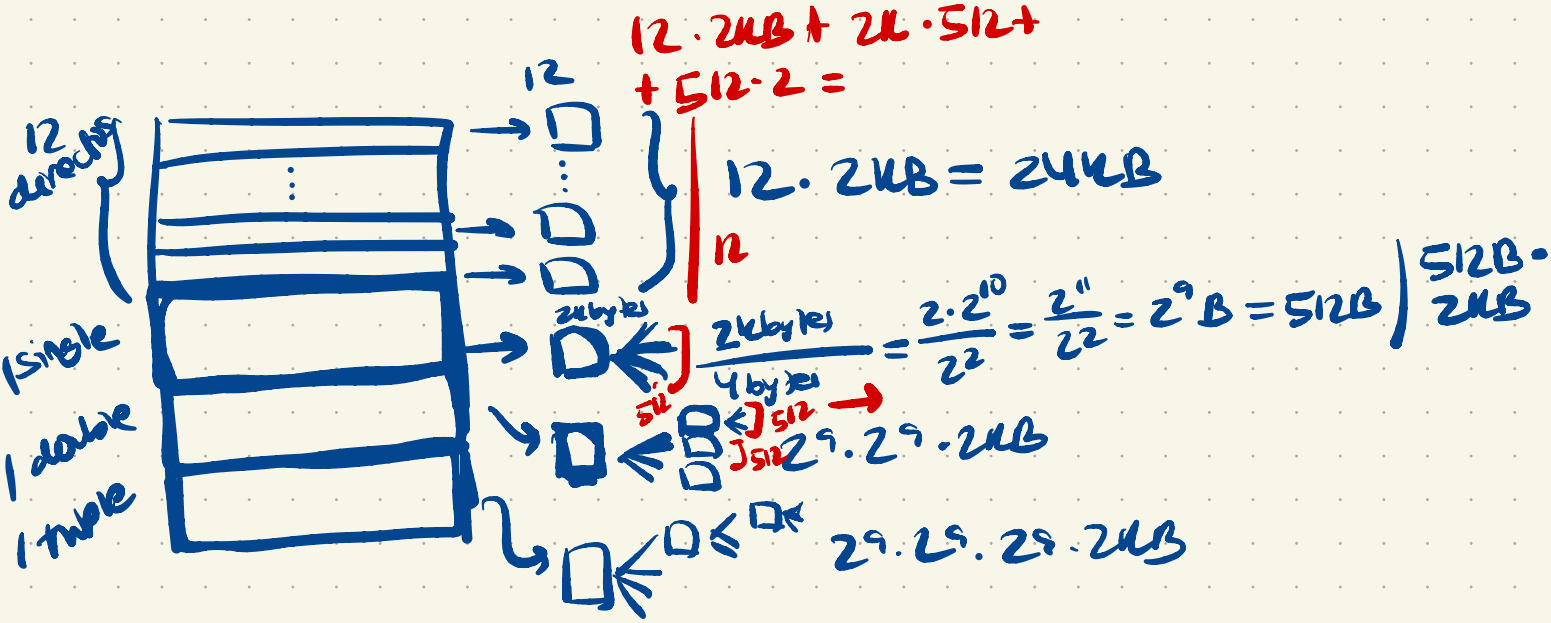
Entradas localizadas en el primer bloque excepto
students → 40, byban → 20



9 accesos + traida de lista de inodos a M.P.

1.2) tbloque = 2Kbytes
12 bloques directos
1 para single, doble y triple.
4 bytes dirección

Bloques de disco necesarios para almacenar fich bytes (6 MBytes) contiros para datos y cuantos para índices.



$12 \cdot 2KB + 2K \cdot 512 + 512 \cdot 2 =$

$12 \cdot 2KB = 24KB$

n

$\frac{2Kbytes}{4bytes} = \frac{2 \cdot 2^{10}}{2^2} = \frac{2^{11}}{2^2} = 2^9 B = 512B$ | 512B · 2KB

$2^9 \cdot 2^9 \cdot 2KB$

$2^9 \cdot 2^9 \cdot 2^9 \cdot 2KB$

$24KB + 2^9 B \cdot 2KB = 24KB + 2^9 \cdot 2 \cdot 2^{10} = 24KB + 1MB$

$\frac{6 MBytes}{t\ bloque} = \frac{6MB}{2KB} = \frac{6 \cdot 1024^2}{2 \cdot 10^{10}} = 3 \cdot 10^{10} = 3KB$

nº bloques datos

7 bloques índices.

1.3

$4194304 = 4 \cdot 2^{20}$

$\frac{4 \cdot 2^{20}}{2KB} = \frac{4 \cdot 2^{20}}{2 \cdot 2^{10}} = 2 \cdot 2^{10}$ bloques de datos =

$= 2 \cdot 1024 = 2048$

$\frac{2 \cdot 2^{10}}{2^7} = 2 \cdot 2^3 = 16B$

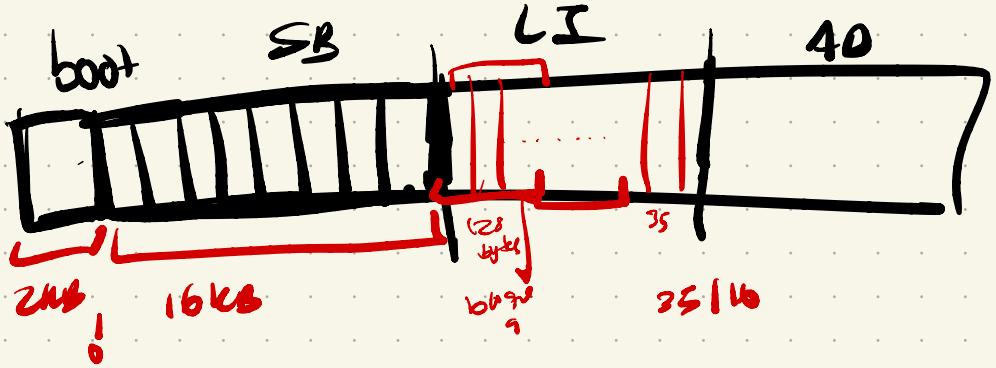
1 bloque = 16 inodos

↓

accede a 3 bloques 2 de índices y uno de datos.

128 bytes

1.4

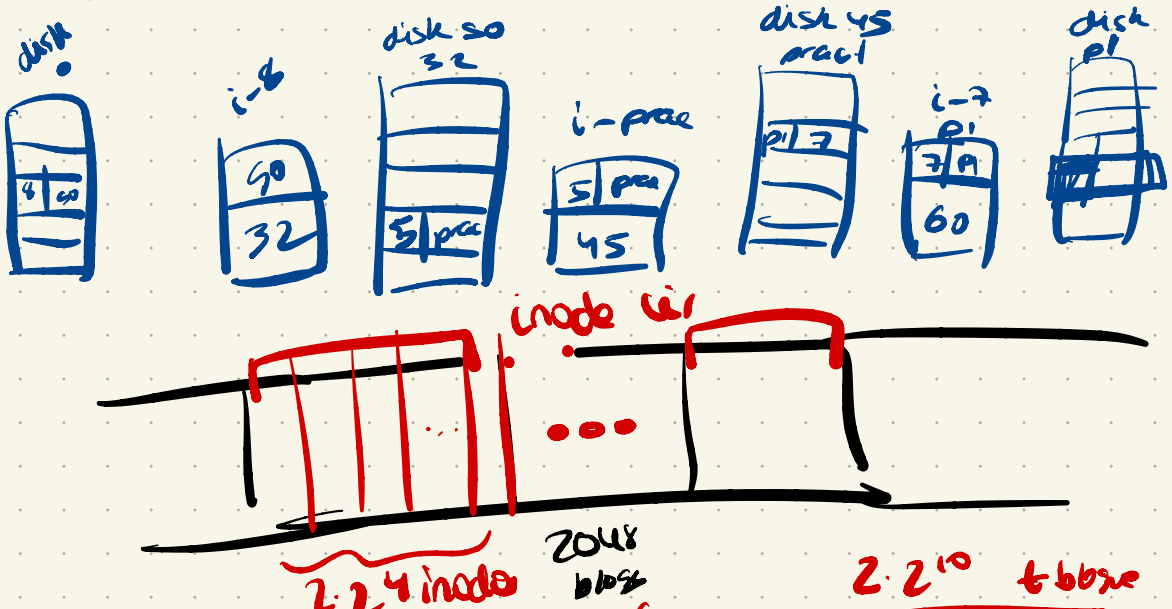


2,5 MB

$$\frac{2{,}5 \cdot 2^{20}}{2 \cdot 2^{10}} = 1{,}25 \cdot 2^{10} = \underline{1{,}25\,KB} = \underline{1280 \text{ bloques}}$$

1280 - 12 - 512 - 512 - 244

1111



$2 \cdot 2^4$ inodos

64 bytes = 2^6 / inodo

2048 = $2^{10} \cdot 2$ bloques

$$\frac{2 \cdot 2^{10} \text{ bloques}}{2^6} = 2 \cdot 2^4 \text{ inodos / bloque}$$

$2 \cdot 2^{10} \cdot 2 \cdot 2^4$ inodos totales = 2^{16} bits = 26 Kbits

2KBytes = 16Kbits

$$\frac{26 \text{Kbits}}{16 \text{Kbits}} = 4 \text{ bloques}$$