SOLUCIONES EJERCICIOS 1, 2, 3, 4, 5 y 8 DE FICHEROS

(1) Para obtener los tamaños máximo y mínimo de los ficheros necesitamos conocer el tamaño de BLOQUE.

La secuencia de Bloques de los ficheros obtenida a partir de la FAT será:

Fichero A - 6, 8, 4, 2, 17, EOF (5 Bloques)
Fichero B - 5, 9, 12, 10, 16, 19, 1, EOF (7 Bloques)
Fichero C - 11, 3, 14, 21, 7, EOF (5 Bloques)

Tenemos que la capacidad de direccionamiento de la FAT está totalmente empleada:

- Si el nº bloque lógico usa 16 bits, cada posición de la FAT se expresa con un número de 16 bits.
- Tengo 16 bits por lo que puedo numerar de $0..(2^{16}-1)$ es decir $\boxed{2^{16} \text{ posiciones}}$

$\Rightarrow$ Puedo tener $2^{16} = 2^6 \cdot 2^{10} = 64 \cdot 1024 = \underline{65536 \text{ nºs bloque}}$

Si el tamaño (capacidad) del disco es = 512 Mbytes

$\Rightarrow \underline{\text{Tamaño Bloque}} = \frac{\text{Tamaño disco}}{\text{Nº Bloques}} = \frac{512 \cdot 1024}{64 \cdot 1024} = \boxed{8 \text{ Kbytes}}$

- Tamaños MAX y MIN DE LOS FICHEROS

Ficheros A y C tienen 5 bloques

Tamaño Min → el fichero ocupa 4 bloques + 1 byte del 5º bloque
$= 4 \times 8 \text{ Kbytes} + 1 \text{ byte} = 32769 \text{ bytes}$
Tamaño MAX → el fichero ocupa los 5 Bloques = 40 Kbytes.

Fichero B tiene 7 bloques

Tamaño MIN → el fichero ocupe 6 bloques + 1 byte = 49153 bytes
MAX - " " los 7 bloques completos = 56 Kbytes

(2) Información de los inodos [redacted]

	1	5	7	140	150	20	33
Punteros Directos	3	20	15	31	4	19	14
	10	21	18	37	32	8	16
Puntero Indirecto	30	7	NULL	28	NULL	33	44

INODO : 2 punteros directos que apuntan a bloques de datos
1 puntero indirecto simple que apunta a bloque de punteros los cuales apuntan a bloques de datos

inodo 1: Bloques de datos: 3, 10
p. ind. (30) → Bloques de dats: 11, 12, 23, 45

inodo 5: P. Directo → Bloques de datos 20, 21
P. ind → (7) → Bloques de datos 36, 5, 22, 41

inodo 7 (bin): P. Directo → Bloques de datos: 15, 18

inodo 140 (dev): P. Directo → Bloques de datos 31, 37
P. ind (28) → " " 9, 26

inodo 150 (publico): P. Directo → Bloques de datos 4, 32

inodo 20 (pepe): P. Directo - Bloques de datos: 19, 8
P. ind (33) - " " 6, 13, 1, 2

inodo 33: indice.txt: P. Directo - Bloques de datos 14, 16
P. ind (44) - " " 17, 24, 25, 27

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	2	EOF	10	32	22	13		6	26	11	12	23	1	16	18	17	24	EOF	8	21

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41
36	41	45	25	27	EOF	EOF				37	EOF				5	9				EOF

42	43	44	45
			EOF

③

punteros de 32 bits
Tamaño de Bloque de 16 bytes } → Caben 4 punteros en cada bloque

1

./PRAC/PRSO1/INTER.C

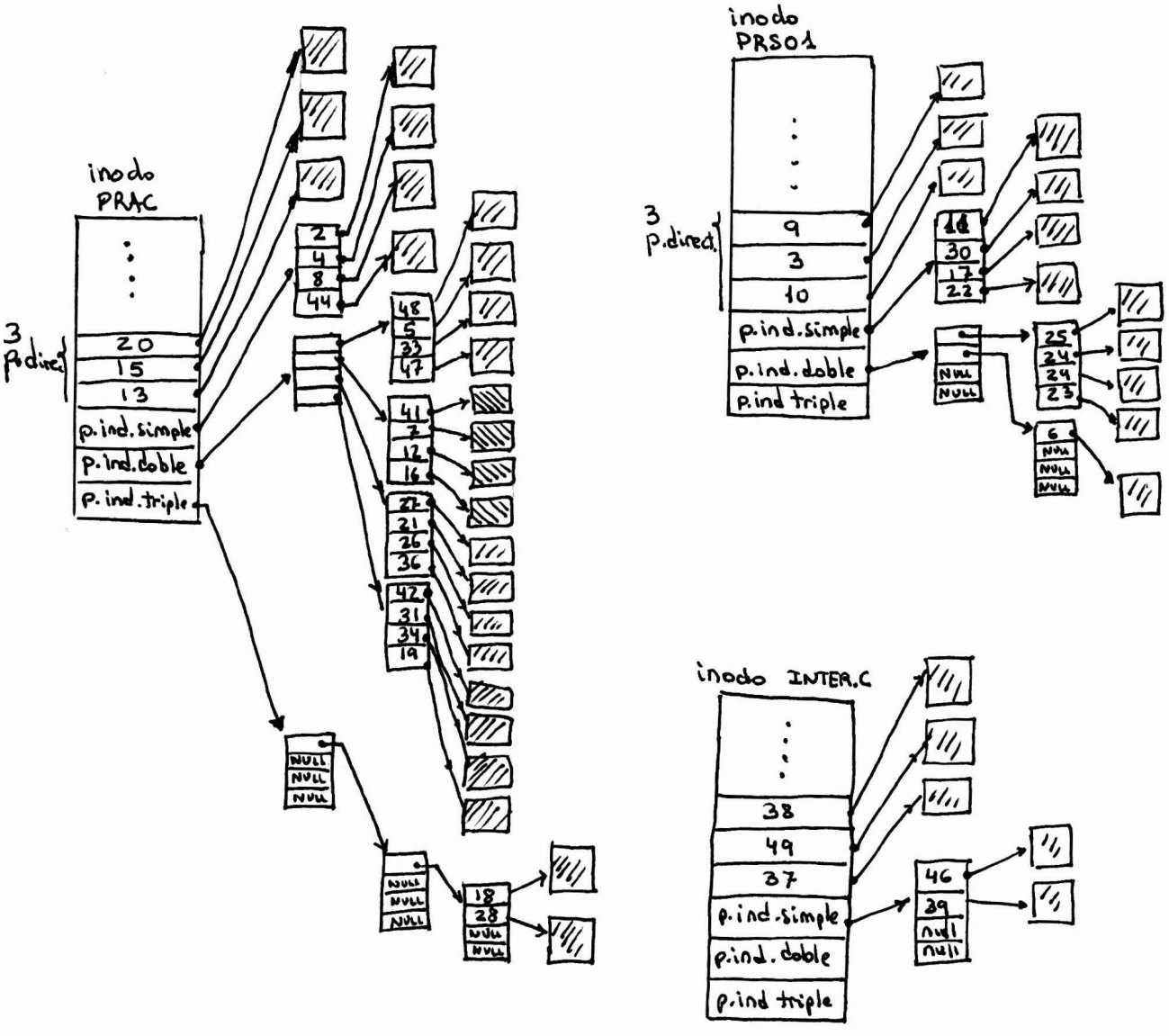
Secuencia de bloque obtenido de la FAT

PRAC: 20, 15, 13, 2, 4, 8, 44, 48, 5, 33, 47, 41, 7, 12, 16, 27, 21, 26, 36, 42, 31, 34, 19, 18, 28, EOF

PRSO1: 9, 3, 10, 11, 30, 17, 22, 25, 24, 29, 23, 6, EOF

INTER.C: 38, 49, 37, 46, 39, EOF

a) Representar lo mismo en UNIX con inodo con 3 punteros directos, 1 indirecto simple, 1 indirecto doble y 1 indirecto triple.



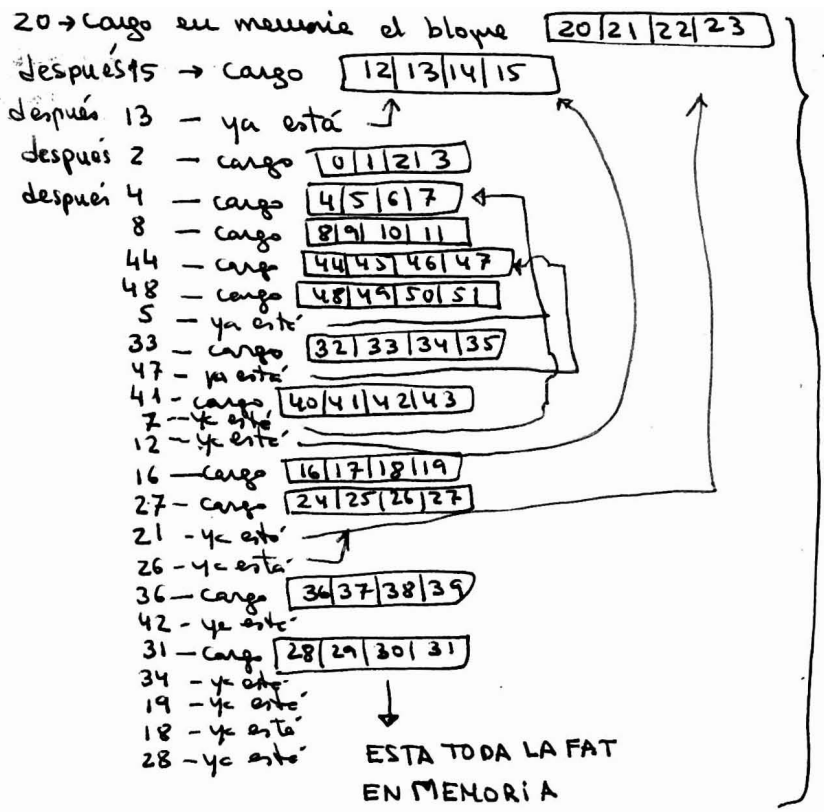
Bloque de datos

3.b.1) ¿Nº MAX de accesos a disco para leer todo el fichero INTER.C?
suponiendo que la FAT no está en memoria.
La ruta del fichero es PRAC/PRSO1/INTER.C

- PRAC es un directorio del actual por lo que podemos encontrar su bloque de entrada en la FAT en la entrada de PRAC en el directoria actual cuyos bloques estarán cargados en memoria
- PRSO1 es un directorio de PRAC para saber en que bloque comienza (9) el sistema operativo debe buscar su entrada en el directorio PRAC, por lo que hay que mirar, en el peor caso, todos los bloques de PRAC
- INTER.C es un fichero cuya entrada estará en el directorio PRSO1, por lo que para saber en que bloque comienza (38) el sistema operativo buscará, en el peor caso, su entrada en todos los bloques de PRSO1.

Tendremos por un lado los accesos a los bloques de la FAT y por otro los accesos para leer los bloques de los directorios y ficheros.

→ Primero leemos la FAT localizando el inicio de PRAC.



→ Como en el peor caso la entrada PRSO1 en PRAC estaría en el ultimo bloque tendré:
13 accesos para leer la FAT
25 accesos para leer los bloques de PRAC
TOTAL = 38 accesos

Hacemos lo mismo con PRSO1 y INTER.C pero solo tengo que leer de disco los bloques, ya que la FAT la puedo leer en memoria
PRSO1 → 12 accesos para leer los bloque (peor caso INTER.C en ultimo bloque)
INTER.C → 5 accesos para leer el fichero INTER.C

TOTAL DE Nº MAX DE ACCESOS = 38 + 12 + 5 = **55 accesos**

3. b2) Usando asignación tipo UNIX, si un inodo son 3 bloques y el inodo PRAC está en memoria.
Suponemos el peor caso, es decir que la entrada buscada este en el último bloque.
Consultamos el esquema del apartado a) podemos contar el nº de bloques al que hay que acceder (de datos y de punteros)

- El inodo PRAC ya está en memoria
  - 3 bloques de datos (punteros directos) → 3 accesos
  - 1 bloque de punteros y 4 bloques de datos (p. ind simple) → 5 accesos
  - 5 bloques de punteros y 16 bloques de datos (p. ind doble) → 21 accesos
  - 3 bloques de punteros y 2 bloques de datos (p. ind. triple) → 5 accesos

  TOTAL accesos en PRAC = 34 accesos

- Se carga el inodo PRSO1 en memoria → 3 bloques → 3 accesos
  - 3 bloques de datos (p. directos) → 3 accesos
  - 1 bloque de punteros y 4 bloques de datos (p. ind. simple) → 5 accesos
  - 3 bloques de punteros y 5 bloques de datos (p. ind dobles) → 8 accesos

  TOTAL accesos en PRSO1 = 19 accesos

- Se carga el inodo INTER.C → 3 Bloques → 3 accesos
  - 3 bloques de datos (p. directos) → 3 accesos
  - 1 bloque de punteros y 2 bloques de datos (p. ind. simple) → 3 accesos

  TOTAL accesos en INTER.C = 9 accesos.

Nº MAX DE ACCESOS = 34 + 19 + 9 = 62 accesos

3. d) ¿TAMAÑO MAX DE UN FICHERO EN UNIX?

A partir del nº de punteros de cada tipo sobre cuantos bloques puede direccionar.

3 punteros directos → 3 Bloques
1 p. indirecto simple → 4 Bloques
1 p. indirecto doble → 4*4 = 16 Bloques
1 p. indirecto triple → 4*4*4 = 64 Bloques

TOTAL = 87 Bloques

TAMAÑO BLOQUE = 16 bytes → **TAMAÑO MAX DE FICHERO = 1392 bytes**

3. e)

¡Tamaño de Bloque! ¡Tamaño de punteros!

¿Que ventajas e inconvenientes?

(4) (6) Sistema que usa FAT.

Los campos de una entrada en un directorio son:

Nombre → 22 bytes
Tipo → 1 bytes
Tamaño → 4 bytes
Nº 1º Bloque → 4 bytes
Permisos → 1 byte

} Sumando:

EL TAMAÑO TOTAL DE UNA ENTRADA = 32 byte

a) Para calcular el tamaño de la FAT necesito saber cuantos bloques hay en el disco para saber el número de punteros que necesito. Para saber cuantos bloques hay en disco necesito saber el tamaño del Bloque.

Sea TBLOQUE = tamaño del Bloque

Sabemos:

1) EJEMPLO1.C → tiene 4 entradas en la FAT ⇒ usa 4 Bloques
Su tamaño real es 12688 bytes → No sabemos la ocupación del último bloque:

Como MIN ⇒ 3 * TBloque + 1 byte ≤ 12688 bytes
Como MAX ⇒ 4 * TBLOQUE ≥ 12688 bytes

Despejando MIN → $TBLOQUE \leq \frac{12688-1}{3} = 4229$ bytes
MAX → $TBLOQUE \geq \frac{12688}{4} = 3172$ bytes

SE DEBE CUMPLIR: $3172 \text{ bytes} \leq TBLOQUE \leq 4229 \text{ bytes}$ ①

2) Sabemos que el directorio EJECUTABLES tiene un tamaño 8192byte que coincide exactamente con un número entero de Bloques.

$Nº. TBLOQUE = 8192 \text{ bytes}$ ②

Combinando las ecuaciones ① y ②

Si Nº = 1 ⇒ ② ⇒ TBLOQUE = 8192 bytes → NO CUMPLE ①

Solución: Si Nº = 2 ⇒ ② ⇒ TBLOQUE = 4096 bytes → CUMPLE ①

Si Nº = 3 ⇒ ② ⇒ TBLOQUE = 2731,5byte → NO CUMPLE ①

EN DISCO CABEN = $\frac{40 \text{Mbytes}}{TBLOQUE}$ = 10240 Bloques

Si tamaño del puntero = 4 bytes

FAT ocupa = 4 * 10240 = 40 Kbytes = 10 Bloques

4b) El Nº MAX y MIN de entradas en EJECUTABLES

¿cuantas entradas caben en un bloque? ¿Cuantos bloques tiene EJECUTABLES?

Tamaño de ejecutables = 8192 bytes = 2 Bloques

Tamaño Bloque = 4096 bytes

Tamaño de una entrada = 32 bytes

MIN Nº entradas →	~~129~~
MAX Nº entradas →	~~64~~ 256

4096 / 32 = 128 entradas en 1 Bloque

4c) Nº accesos máximo y mínimo para ejecutar Almacen.x?

En memoria están los bloques del directorio padre, por lo que puedo leer la entrada de nuestro directorio sin acceder a disco, es decir

¡ Bloque de comienzo de nuestro directorio !

Como el directorio tiene 6 entradas suponemos que solo usa 1 BLOQUE

SUPONEMOS QUE LA FAT NO ESTÁ EN MEMORIA

- Acceso en FAT al índice del bloque del directorio y veres EOF → 1 Acceso FAT
- Leo el bloque → 1 acceso
  y encuentro la entrada de almacen.x y sé que su primer bloque es 2208
- Accedo a la posición de la FAT 2208 y obtengo el resto de índices de bloque del fichero. almacen.x

Tamaño almacen.x = 4106404 bytes → divido entre Tamaño de bloque → 1003 Bloques

⇒ 1003 punteros en la FAT ocupan → 4012 bytes

En el mejor caso cabrían en 1 Bloque de la FAT, en el peor en bloques diferentes

Mejor caso: los punteros del fichero en la FAT todos en el mismo Bloque → 1 acceso en FAT para encontrar todo el fichero

Peor caso: los punteros en bloques diferentes → Los 10 Bloques que ocupan la FAT → 10 accesos

- Hay que leer los 1003 Bloques de datos → 1003 accesos

Direc { FAT 1 acceso; Bloque 1 acceso }

Fichero { FAT MIN 1 acceso; MAX 10 accesos; 1003 accesos }

⇒ MAX = 2 + 1003 + 10 = 1015 accesos

MIN = 2 + 1003 + 1 = 1006 accesos

4 d) UNIX

inodo con 512 p. directos, 64 indirectos, 32 dobles indirectos

los punteros de 4 bytes → en 1 bloque $\frac{4096}{4}$ = Caben 1024 punteros

- El directorio padre cargado en memoria ⇒ puedo leer el número de inodo de nuestro directorio sin acceder a disco.

- Suponemos que el inodo ocupa solo un bloque ya que todos los punteros caben en un bloque y no se nos da información sobre el resto de parámetros del inodo.

Directorio
- Para cargar el inodo del directorio (1 Bloque) → 1 acceso a disco
- 1 acceso al bloque del directorio

Fichero
- localizamos le entrada del fichero → un acceso para cargar el inodo de ALMACEN.K

El fichero ocupa 1003 Bloques
- p. directos: 512 Bloques → 512 accesos
- p. indirectos: con 1 puntero accedo a 1024 punteros (1 Bloque) que apuntan a 1024 Bloques

1003 − 512 = 491 Bloques

1 acceso al bloque de punteros

491 accesos a los 491 bloques restantes del fichero

$$\text{TOTAL} = \underbrace{(1+1)}_{\text{direct.}} + \underbrace{(1+512+1+491)}_{\text{fichero}} = 1007 \text{ accesos}$$

como mínimo ya que hemos supuesto que el inodo solo es un bloque.

5) Directorio raiz del dispositivo bloques 3, 4, 5 y 6

Los Bloques del directorio raiz contienen 4 entradas que son:

1) pepe.obj : 9, 32, 16, 21, 26, 8, 10, EOF
   KK.c : 18, 11, 29, 15, 7, 21, 26, 8, 10, EOF
   hola.c : 30, 17, 13, 20, 17, 13, 20 ... se repite
   pra1.os : 27, 31, 19, EOF

2) Errores:
   a) – pepe.obj y KK.c tienen 4 bloques comunes
   b) – hola tiene un bucle
   c) – en la FAT quedan indices a bloques que no pertenecen a ninguno de los ficheros

3) Para solucionar cualquier problema hace falta al final la colaboración del usuario que decidirá que hacer.

   a) solución: en p.ejemplo el fichero KK.c copiar los 4 bloques 21, 26, 8, 10 en otros 4 libres de la FAT y ajustar la nueva secuencia

   El usuario despues decidira si esos bloques eran uno de uno u otro fichero.

   b) hola.c debe acabar en el bloque 20

   c) los indices no usados se deben convertir en una o varias secuencias por si la información es de utilidad p.ejemplo que sean ficheros perdidos
   Después el usuario decidirá que hacer con ellos.

(8)

	Primer bloque	Tamaño
SOL	17	2700 bytes
Examen	7	1603 bytes
Junio 98. doc	54	1050 bytes

a) Nos aseguran que el tamaño del bloque del disco es 102 bytes, ¿ES cierto?

SOL : 17, 33, 9, 23, 3, 30, 49, 46, 43, 25, 5, 13, 20, 35
42, 51, 8, 22, 36, 47, 18, 26, 52, 39, 19, 11, 32, EOF

Examen: 7, 21, 38, 31, 15, 27, 48, 24, 50, 41, 55, 29, 16, 6
34, 12, 40, EOF

Junio 98. doc : 54, 2, 10, 1, 14, 28, 45, 37, 4, 44, 53, EOF

1) SOL → 27 Bloques → MAX = 27 Bloques * 102 bytes ; MIN = 26 Bloque * 102 + 1 byte
2) Examen: 17 Bloques → MAX = 17 Bloque * 102 byte ; MIN = 16 Bloque * 102 + 1 byte
3) Junio 98. doc: 11 Bloques → MAX = 11 * 102 byte ; MIN = 10 Bloque * 102 + 1 byte

1) ⇒ [illegible]
$27 * 102 = 2754 \geq 2700 \text{ bytes} \geq 2653$
2) ⇒ $17 * 102 = 1734 \geq 1603 \text{ bytes} \geq 1633$ ⇒ NO SE CUMPLE para Examen
3) ⇒ $11 * 102 = 1122 \geq 1050 \geq 1021$

⇒ El tamaño del Bloque no puede ser 102 bytes

8 b) En un bloque caben 6 direcciones de bloque

d Max y Min nº de accesos a bloque de disco para leer Junio.doc?

b.1) FAT no cargada en memoria.

Sol / Examen / Junio98.doc

Leer la FAT sol:

17 Bloque | 12 | | 17 |
33 Bloque | 30 | | 35 |
9 Bloque | 6 | | 11 |
23 | 18 | | 23 |
3 | 0 | | 5 |
30 ya esta
49 | 48 | | 53 |
46 | 42 | | 47 |
43 ya esta
25 | 24 | | 29 |
5 ya esta
13 ya esta
20 ya esta
35 ya esta
42 ya esta
51 ya esta
8 ya esta
22 "
36 "
47 "
18 "
26 "
52 "
39 Bloque | 38 | | 41 |
19 ya esta
11 "
32 "

↓

1 acceso

FAT Examen:

~~55~~ → bloque | 54 | 55 | - - -

los demas estan

↓

1 acceso

FAT Junio 98.doc

ya esta toda la FAT en memoria

↓

0 accesos

- Para leer Junio 98.doc primero se leen los Bloques de sol hasta encontrar la entrada Examen

  MIN encontrarla en el 1º Bloque → 1 acceso

  MAX tener que leer todos → 27 accesos

- Después hay que leer los Bloques de Examen hasta encontrar la entrada de Junio98.doc

  MIN encontrarla en el 1º Bloque – 1 acceso

  MAX leer todos → 17 accesos

- Por último leer todos los Bloques del fichero Junio98.doc

  → 11 accesos

Total

sol: FAT = 9 accesos
MIN Bloque = 1 acceso
MAX " = 27 accesos

Examen: FAT = 1 acceso
MIN Bloque = 1 acceso
MAX Bloque = 17 accesos

MINIMO: 9 accesos + 1 + 1 + 1 + 11 = 23 accesos

MAXIMO: 9 accesos + 27 + 1 + 17 + 11 = 65 accesos

8 b.2) Unix 3 punteros directos, 1 puntero ind. simple
1 puntero indirecto doble

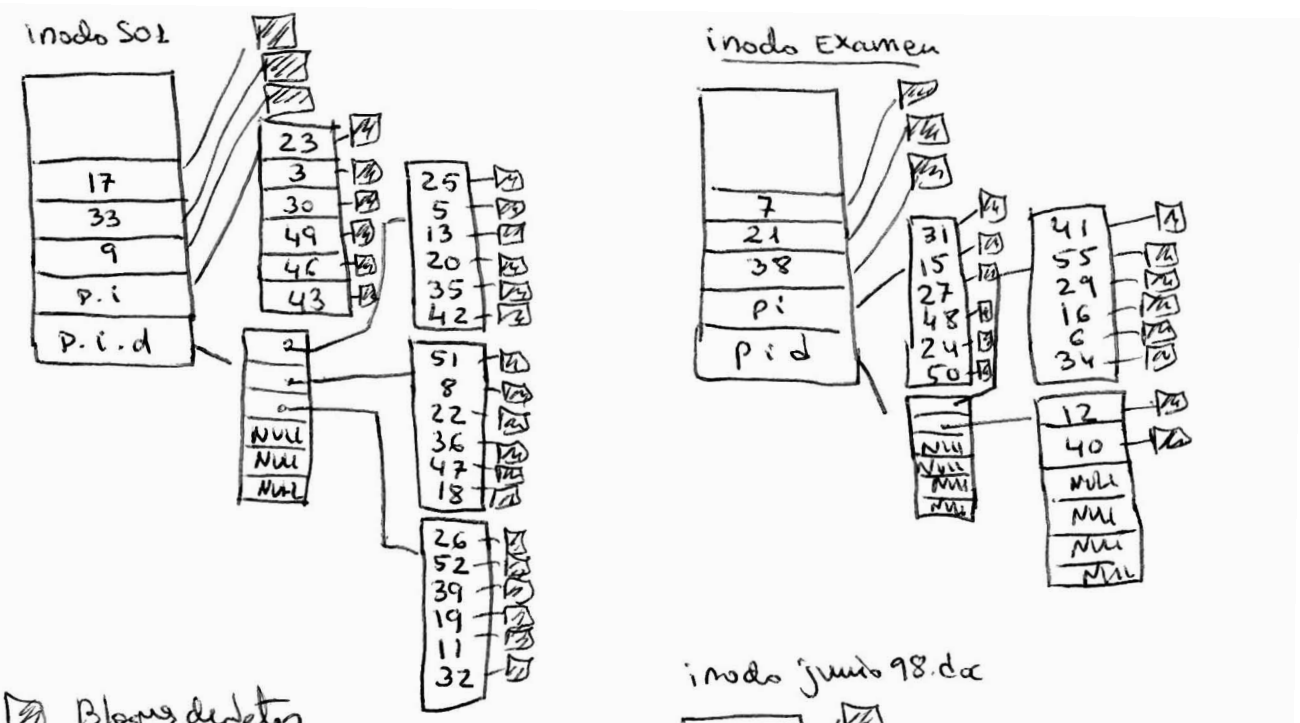
inodo SO1 en memoria

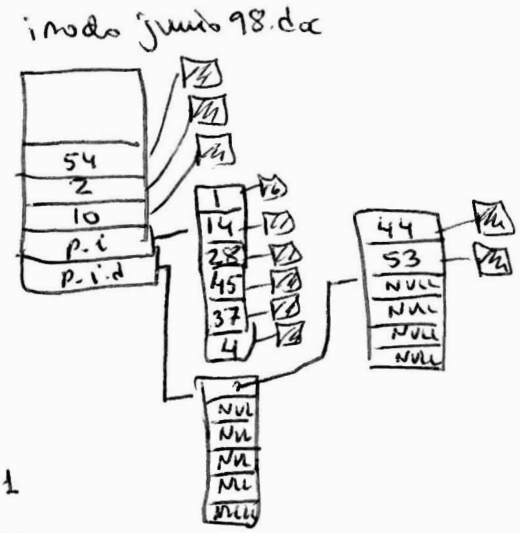
En un bloque caben 4 inodos.

Para leer los inodos si los 3 están en el mismo bloque ya estarían en memoria ⇒ MIN 0 accesos

Si cada inodo en un bloque diferente necesito

MAX → 2 accesos





SO1

P.Directos – (3) acceso (dato)
(1) acceso (p.i) + (6) acceso (dato)
(4) acceso (p.i.d) + (18) acceso (dato)

→ MIN = 1
→ MAX = 32

Examen

P.Directos – (3) acceso (dato)
(1) acceso (p.i) + (6) acceso (dato)
(3) acceso (p.i.d) + (8) acceso (dato)

MIN = 1
MAX = 21

junio 98.doc

P.Directo → (3) acceso (dato)
(1) acceso (p.i) + (6) acceso (dato)
(2) acceso (p.i.d) + (2) acceso (dato)

MAX = 14
MIN

TOTAL

MIN = 0 + 1 + 1 + 14 = 16

MAX = 2 + 32 + 21 + 14 = 69