

# Sistema de Fitxers

**1.-** En un sistema de ficheros que utiliza una tabla global de encadenamientos (FAT) la configuración de cada entrada del directorio es:

{ Nombre\_del\_fichero, tipo, dirección\_del\_primer\_bloque }

Sabemos que el tamaño de la unidad de asignación (cluster) es igual al tamaño de la unidad de transferencia (bloque). Se pide:

a) Dibujar el árbol del directorio para los valores de bloques y FAT expuestos en la figura

FAT	bloque raíz	bloque 2	bloque 3
2 eof	B dir 3	G dat 14	D dat 4
3 eof	H dat 10	P ? 11	E dat 6
4 eof	C dir 2	R dat 12	F dat 7
5 eof	D dir 13	:	:
6 5	:	:	:
7 9			
8 eof	bloque 11	bloque 12	bloque 13
9 8	/B/F	/B/E	
10 eof			
11 eof			
12 eof			
13 eof			
14 eof			

- Suponiendo que toda la FAT está siempre en memoria, ¿Cuántos accesos a disco se precisan para leer el tercer bloque del fichero F?, ¿y el cuarto?
- De que tipo(s) puede ser el fichero P. Razónalo.
- Si se supone que utilizamos las llamadas al sistema de ficheros definidas en la práctica, qué valores deben tomar  $r$ ,  $x$ ,  $y$ ,  $v$  en las siguientes llamadas:

`fd = open(x, O_RDONLY, 0);`

`lseek(fd, r, y);`

`read(fd, z, v);`

sabiendo que después de estas llamadas  $z$  queda con el valor "/E". ¿Cuántos bytes de datos se han transferido del disco a memoria?. ¿Y al usuario?.

**2.-** Donat el següent sistema de fitxers basat en FAT, quants accessos a disc serien necessaris (especifiqueu quins) per obtenir el 4rt bloc de dades del fitxer /A/S? Considereu que la FAT no està carregada a memòria i que sabeu que el directori arrel es troba al bloc de dades 2

FAT:.

2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
eof	eof	16	eof	eof	eof	eof	eof	eof	eof	13	4	eof	eof	6	eof

.	dir	2	.	dir	7	/B/R	.	dir	9
..	dir	2	..	dir	2		..	dir	2
A	dir	7	S	dat	8		R	dat	12
B	dir	9					T	dir	15
2			7			8	9		

### 3.- (parcial 2005 QT) Donat el següent sistema de fitxers UNIX:

#### Inodes

tipus	dir	dir	dir	link	dat	dat
blocs de dades	.....	.....	.....	.....	.....	.....
núm i-node	3	5	4,6	1		2
	2	3	4	5	6	7

L'inode 2 es l'arrel

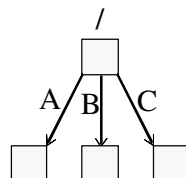
#### Blocs de dades

/b/c	abcdefg hi	.	2	.	4	.	3	c	8
		..	2	..	2	..	2		
		a	3	a	6	a	7		
		b	4	b	7	b	5		
1	2	3	4	5	6	7	8		

Suposant que no hi ha cap buffercache i que el superbloc està sempre a memòria. Quins accessos faran al disc les crides a sistema següents. Suposem que no hi ha cap altre procés executant-se al sistema.

```
fd = open("/a/b", O_RDONLY);
read(fd, buffer, 100);
fd = open("/b/d", O_WRONLY);
```

### 4.- Tenemos un sistema de ficheros como el de la figura:



Ejecutamos una serie de comandos, cuyo resultado en la pantalla es:

```
$ pwd
/A
$ echo "estoy en el directorio A" > D
$ ln -s /A /A/E
```

```
$ ln -D /A/F
```

Donde *ln par1 par2* crea un hard-link, ó un soft-link ( opción *-s* ) de *par2* a *par1*.

Se pide:

a) Dibuja el nuevo sistema de ficheros con los cambios introducidos.

b) Cuál sería el contenido del fichero A?

c) Cuál sería el contenido de los ficheros D, E, y F?

d) Si ahora ejecutáramos:

```
$ mkdir G
```

```
$ ln -f /A /A/G/H /* opción -f: fuerza hard-link entre directorios (sólo super-user */
```

Cuál sería el contenido de los ficheros A y G?

e) Qué problemas se podrían encontrar en un sistema de ficheros como el del punto d)?

**5.-** En un sistema de fitxers de tipus UNIX ens queden encara 200 blocs de dades lliures, però en canvi no podem crear cap més fitxer. Quin motiu pot provocar això?

**6.-** Encuentra las dos inconsistencias de este sistema de ficheros de UNIX y responde a las siguientes preguntas..

i_node 5	i_node 6	i_node 7	i_node 9	
dir	dir	dir	dir	Tipo de fichero
6	6	9	4	Tabla de índice (1 <sup>er</sup> bloque)

bloque 4	bloque 6	bloque 9
.	.	.
..	..	..
A	C	E
B	D	F
11	5	7
4	4	9
7	7	11
17	8	14

a) ¿Por qué en el bloque 9, el número de i\_node de “..” es también 9?

b) ¿Por qué el primer bloque en los i\_nodes 5 y 6 es común?

c) ¿Por qué el i-node 7 aparece en dos directorios?

d) ¿Por qué “C” es de tipo “link”, y en el i\_node pone que es de tipo “dir”?

**7.-** Donat el següent sistema de fitxers tipus Unix, indica les 3 inconsistències:

nom fitxer	/	A	B	D	C	L
tipus	dir	dir	dir	dat	link	dat
	.....	.....	.....	.....	.....	.....
blocs de dades	10	22	6	9	15	9
núm. links	3	3	2	1	3	3
núm i-node	3	4	7	11	18	21

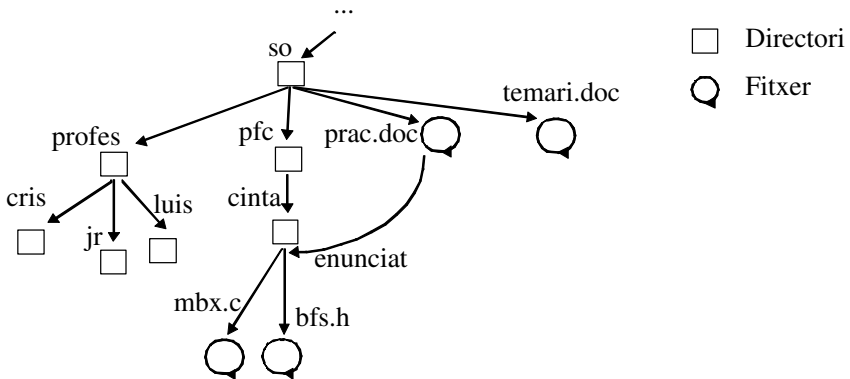
.	7	Please allow me to introduce myself; Why I'm a man of wealth and taste; I've been around for a long, long year; Stole many a man's soul and faith	.	3	/A/B/D	.	4
..	4		..	3		..	3
C	18		A	4		B	7
D	11		B	7		C	18
			C	18		L	21
6		9	10		15	22	

**8.-** Quin és l'error que presenta aquest sistema de fitxers FAT?

ROOT			bloc 1			bloc 10		
A	dir	1	x	dat	2	z	dat	11
B	dir	10	y	dat	5			

FAT												
índex	2	3	4	5	6	7	8	9	10	11	12	13
EOF	3	EOF	Free	6	8	10	EOF	Free	EOF	12	EOF	Free


**9.-** Tenim un sistema de fitxers amb un espai de noms en graf, amb els següents directoris i fitxers:




Sabent que l'organització emprada és la de UNIX, dir quins errors apareixen a les següents estructures


Nom Fitxer	profes	pfc	jr	prac.doc	temari.doc	cris	luis
Tipus de fitxer	dir	dir	dir	link	dades	dir	dir
Num. opens	5	3	2	2	2	2	2
...							
Índex al primer bloc	60	63	62	69	63	61	66
Número d' i_node	3	4	5	6	7	8	9

### Blocs de Dades:

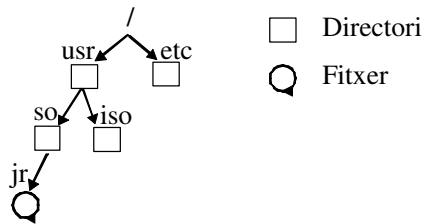
nom	Num bloc
..	2
cris	8
jr	5
luis	9
	

bloc 60

nom	Num bloc
..	10
mbx.c	51
bfs.h	31
enunciat	6
	
bloc 62	

Directori SO	
nom	Num bloc
..	27
profes	3
temari.doc	7
pfc	4
prac.doc	6
temari.doc	7
 bloc 64	

**10.-** Tenim un sistema de fitxers amb un espai de noms en arbre, amb el següents directoris, subdirectoris i fitxers:



Sabent que l'organització emprada és la de UNIX, omplir els camps buits amb les dades que falten:

Taula d' i\_nodes

Tipus de fitxer

dir

data

Índex al primer bloc

5

6

2

1

4

Número d' i\_node

2

3

4

5

6

7

Blocs de Dades

Hola!

.

..

.

..

2

.

..

2

.

..

3

2

.

..

3

2

7

.

..

jr

3

6

1

2

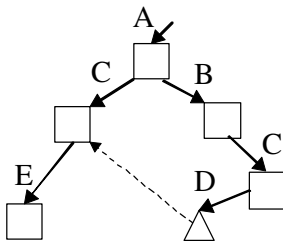
3

4

5

6

**11.-**



El fitxer D és de tipus link. El contingut del fitxer D és /A/C  
 En memòria tenim el super-bloque.  
 En memòria tenim espai suficient per mantenir tot el que se  
 traiga de disc.

- Quants accessos a disc fan falta per llegir el bloc 15 de D?

**a)** Estructura de i-nodes. Cada i-node ocupa un bloc. El directori A se troba en el i-node 2 (informació del super-bloque).

**b)** FAT. La FAT ocupa un bloc. El directori A se troba en el bloc 4 (informació del super-bloque).

**c)** Observas algun problema en aquest sistema de fitxers?

**12.-** En un sistema de ficheros tipo Unix, tenemos los i-nodes y bloques de datos con el siguiente contenido:

I-nodes		2	4	5	6	7	9
	Tipó	dir	link	dir	dat	link	dat
	Índices	9	10	11	4 12 6 5	8	13 7

Bloques	8	9	10	11		
	/B	.	2	/B/D	.	5
		..	2		..	2
		A	7		C	4
		B	5		D	6
	F	9		E	9	

**1) Suponiendo que:**

- Tenemos cargado el super-bloque en memoria.
- No tenemos ningún i-node en memoria. Cada i-node ocupa 1 bloque.
- En memoria sólo disponemos de una **buffer-cache de 6 entradas** que nos ahorra accesos a disco.

**a)-** Dibujar el árbol de directorios correspondiente a este sistema de ficheros, si sabemos que el I-node 2 corresponde al directorio raíz.

**b)-** ¿Cuántos accesos a disco son necesarios para leer el 3r bloque del fichero /A/C, si la buffer-cache utiliza un algoritmo de reemplazo LRU (Last Recently Used) si sabemos que no está siendo usado por ningún otro proceso (es decir, primero se hace una operación de abrir el fichero y luego la de leer)?

**2) Suponiendo que:**

- Tenemos cargado el super-bloque en memoria. Sabemos que el directorio raíz se encuentra en el bloque 9.
- Nuestro sistema de ficheros es como el DOS pero que admite links, y en los bloques de tipo dir se introduce en cada entrada el tipo de fichero, y se cambia el n° de I\_node por el índice del primer bloque.
- La FAT ocupa 1 bloque, y no se encuentra cargada en memoria.
- En memoria sólo disponemos de una **buffer-cache de 3 entradas** que nos ahorra accesos a disco.

**c)-** Dibujar cómo quedaría la FAT.

**d)-** ¿Cuántos accesos a disco son necesarios para leer el 3r bloque del fichero /A/C, si la buffer-cache utiliza un algoritmo de reemplazo LRU y si sabemos que no está siendo usado por ningún otro proceso?

**13.-** Quants i quins accessos són necessaris per llegir el 5é bloc de dades del fitxer /asig/SO/final.txt en aquest sistema de fitxers amb assignació contínua? Podeu assumir que el 5é bloc de dades existeix

ROOT		
Asig	Dir	1
Prac	Dir	23

bloc 1		
SO	dir	7
ISO	dir	9

bloc 7		
final.txt	dat	11

**14.-** En un sistema de fitxers tipus Unix, trobem i-nodes i blocs de dades amb el següent contingut:

I-nodes		2	4	5	6	7	9	10	11	12	13
Tipus		dir	dir	dir	link	dir	dir	link	dat	dir	dat
Índexos		9	8	10	7	13	11	17	14 15 16	12	4 5 6

Blocs		7	8	9	10	11	12	13	17
		/B/D	. 4 .. 7	. 2 .. 2 A 7 B 5	. 5 .. 2 D 9 s 11	. 9 .. 5 H 12 q 13 r 10	. 12 .. 9	. 7 .. 2 C 4 E 6	/B/s

Es demana:

- Dibuixar l'arbre de directoris corresponent a aquest sistema de fitxers, sabent que l'i-node 2 correspon al directori arrel.
- Calcular el nombre d'accessos a disc necessaris per a obrir /A/E/r, tenint en compte que l'i-node arrel sempre està en memòria i que només disposem d'un buffer de tamany 1 bloc i el fitxer no està en ús per cap altre procés, és a dir, primer heu d'obrir el fitxer i després llegir el bloc. Indiqueu què és el que llegiu en cada accés.
- Dibuixeu com quedaria la FAT en un sistema de fitxers que la utilitzi, suposant que els blocs de dades i els directoris estan al mateix lloc que en el cas de Unix.
- Si esborrem el fitxer /B/D/r, indiqueu quins blocs de dades es veuen afectats en el cas de Unix i en el cas amb FAT. Indiqueu també com quedaria la FAT.



**15.-** Es disposa de la següent llista d'inodes i blocs de dades, en un sistema de fitxers basat en taula d'índexos

I-nodes		2	3	4	5	6	7	8	9	10	11
Tipus		dir	data	dir	link	data	data	dir	dir	dir	data
Índexos		2	11	9	10	7	8	5	6	3	4

Blocs		2	3	4	5	6	7	8	9	10	11
		. 2	. 10	abc	. 8	. 9	ijk	rst	. 4	./fff	xyz
		.. 2	.. 4		.. 4	.. 8			.. 2		
		A 4	fff 5		C 10	fff 3			B 8		
		fff 6			D 9				E 10		
					fff 7				fff 11		

Assumint que cada bloc ocupa un sector i que es disposa a memòria d'un buffer de tamany il·limitat, inicialment sense cap informació útil, i que es coneix que l'inode arrel és el número 2, es demana que responeu a les següents preguntes:

- a) Quants accessos a disc es necessiten per accedir al primer bloc de dades del fitxer /A/B/C/fff si no hi ha cap altre procés accedint a aquest fitxer en aquell moment? Indiqueu, per cada accés a disc, què és el s'està llegint.
- b) Quin és el contingut del fitxer llegit?

**16.-** Donat el següent conjunt d'inodes i blocs de dades en un sistema de fitxers Unix, en el qual es permeten fer *Hard Links* entre directoris:

Inodes:		2 (root)	5	7	8	9	11	15
(tipus fitxer)		DIR	DIR	?	?	Fitxer	DIR	Fitxer
(index 1er BD)		1	3	5	2	6	4	7

Blocs de Dades:		1	2	3	4	5	6	7
		. 2	. 8	. 5	. 11	/A/./B/f2	...	...
		.. 2	.. 5	.. 2	.. 5			
		A 8	f1 7	B 11	f2 9			
		B 5		C 8				
				f2 15				

- a) Quina de les següents afirmacions és certa (marqueu-ne només una)?
- Els inodes 7 i 8 podrien ser de tipus *Fitxer Ordinari*.
  - L'inode 7 ha de ser de tipus *Link* (soft-link) i l'inode 8 ha de ser de tipus *Director*.
  - L'inode 7 pot ser de tipus *Link* (soft-link) i l'inode 8 pot ser de tipus *Hard Link*.
  - Totes les respostes anteriors són certes.

Assumiu que l'inode 7 és de tipus *Link*, i que el sistema de fitxers especificat és consistent i complert. Si sabem que l'inode arrel és el número 2, suposant que disposem a memòria d'una buffer cache (inicialment buida) del tamany d'un sector, i que els inodes i els directoris ocupen un bloc i que els fitxers implicats no estan sent accedits per cap altre procés en aquell moment, responeu a les següents qüestions:

**b)** Quants accessos a disc es necessiten per a accedir al 5é bloc del fitxer */B/C/f1*?

- 10
- 17
- 19
- Cap de les respostes anteriors és certa.

**c)** A l'apartat (b) s'accedeix al 5é bloc del fitxer *f2*. Quin és el mínim número d'accessos a disc que es necessita, a partir del directori arrel, per accedir a aquest mateix bloc (és a dir, sense passar per el soft-link)?

- 6
- 7
- 8
- Cap de les respostes anteriors és certa.

**17.-** Tenim un sistema de fitxers amb un espai de noms en graf, i amb organització basada en i-nodes (igual que en Unix). Donat el següent grup d'inodes i blocs de dades:

<b>Inodes:</b>	<b>2 (root)</b>	<b>4</b>	<b>8</b>	<b>15</b>	<b>23</b>	<b>30</b>	
	dir	dades	dir	dades	dir	link	<i>tipus</i>
	...	...	...	...	...	...	
	1	4	23	15	8	5	<i>1er bloc</i>
	4	1	3	1	2	1	<i>#links</i>

<b>Blocs de Dades:</b>	<b>1</b>	<b>8</b>	<b>23</b>	<b>4</b>	<b>5</b>	<b>15</b>
	. 2	. 23	. 8	abc	/B/C/f2	xyz
	.. 2	.. 2	.. 2			
	A 8	C 8	f1 30			
	B 23		f2 4			
	f2 15					

**a)** -Assumint que disposem de un buffer a memòria de tamany il·limitat (inicialment buit), que cada bloc de dades i cada inode ocupa un sector, i que coneixem el nombre del sector en el que es troba el inode arrel, indiqueu **quants i quins accessos** a disc es necessiten per a llegir al primer bloc del fitxer */A/f1*. Cap altre procés està accedint a aquest fitxer en aquell moment, de manera que cap informació es troba a la taula de fitxers estàtica i heu d'obrir el fitxer abans de llegir el bloc de dades.

**b)** -Quina diferència hi ha, a un inode, entre el camp *tipus* (quan aquest és de tipus *link*) i el camp *#links*? **Raoneu breument** la vostra resposta.

**18.-** Dado el siguiente grupo de i-nodos y bloques de datos en un sistema de ficheros de Unix

I-nodos:

Dir	link	dir	link	dat	dir	link	dat
.....	.....	.....	.....	.....	.....	.....	.....
3	8	6	13	10	11	5	26
				4			
				16			
2	4	9	11	14	15	16	19

Bloques de Datos:

.	2	with a sack on his back, And he put down his load where he thought	../D/F	.	9	/C/L
..	2			..	15	
A	15			M	11	
E	4			F	14	
C	9			L	19	
3	4	5	6	8		

A long time ago came a man on a track, Walking thirty miles	.	15	/	it was the best, He made a home in the wilderness.	/C/L
	..	2			
	G	11			
	R	4			
	T	16			
	D	9			
10	11	13	16	26	

a) Dibujar el grafo de ficheros correspondiente. ¿Puede haber algún problema con este grafo de ficheros?

b) Si actualmente estamos en el directorio A, ¿qué diferencias a nivel de accesos hay entre ejecutar los siguientes comandos

\$ ls /

\$ ls ..

\$ ls G

si consideramos que **únicamente** el i-nodo de /A y el bloque de datos de /A se encuentran en memoria? (Nota: El sistema no ha guardado en memoria donde se encuentra el directorio /)

c) Considerando las estructuras internas de UNIX, cuáles y de qué manera se verían afectadas si realizamos las siguientes operaciones (de los i-nodos y bloques de datos que se acceden, sólo comentar aquellos que se modifiquen): (Nota: Son comandos independientes)

\$ rm /C/L (Nota: Considerad que nadie está trabajando actualmente con ese fichero)

fd = open("/A/R", O\_RDONLY, 0644);

**19.-** ¿Cuántos y qué bloques se modifican al hacer un move de un fichero entre dos directorios diferentes, en un sistema con FAT? ¿Y si, por razones de seguridad hubiera 3 copias de la FAT?

**20.-** Es disposa de la següent llista d'i-nodes i blocs de dades en un sistema de fitxers basat en taula d'indexos (Nota: les entrades a 0 d'un directori indica que l'entrada ha estat esborrada):

I-nodes:							
		dir	dir	dir	link	dat	dat
^BDs		2	8	13,9	6	18,24	4,45
#i-node		2	8	9	18	19	23

Blocs de Dades	.	2	/C		H	19	.	8	.	9	Hola	
	..	2			?	0	..	2	..	2		
	A	8					E	23	G	20		
	B	18					F	19	?	0		
	C	9							?	0		
									?	0		
#BD			2	6	9		8		13		18	

a) Dibuixa el graf corresponent a aquest sistema de fitxers.

b) Quins accessos cal fer a disc, i quin és el graf d'aquest sistema de fitxers després de fer els accessos següents si disposeu d'una buffer cache de tamany 1 i que el superblock sempre és a memòria:

```
fd = open("/c/h", O_WRONLY|O_APPEND|O_CREAT, 0600);
unlink("/a/f"); // Esborra el fitxer /a/f
unlink("/c/h"); // Esborra el fitxer /c/h
close(fd);
```

Per indicar els accessos, cal indicar si és un accés a i-node o a bloc, i el número corresponent. Per exemple, I8, B7, indica llegir l'i-node 8 i després el bloc 7 indicant aquelles que es modifiquen (ficant-les en un cercle).

## 21.-(Final 2005 QT) Donat el següent sistema de fitxers tipus UNIX:

Inodes

tipus	dir	data	link	dir	dir
tamany	4096	200	8	6000	4096
BDs	2	4	6	5 3	7
	2	3	4	5	6

Blocs dades

<table><tr><td>.</td><td>2</td></tr><tr><td>..</td><td>2</td></tr><tr><td>etc</td><td>6</td></tr><tr><td>tmp</td><td>5</td></tr></table> <p>2</p>	.	2	..	2	etc	6	tmp	5	<table><tr><td>c</td><td>4</td></tr><tr><td>d</td><td>10</td></tr><tr><td colspan="2"></td></tr></table> <p>3</p>	c	4	d	10			<table><tr><td colspan="2">/tmp/a</td></tr><tr><td colspan="2"></td></tr></table> <p>4</p>	/tmp/a				<table><tr><td>.</td><td>5</td></tr><tr><td>..</td><td>2</td></tr><tr><td>a</td><td>9</td></tr><tr><td>b</td><td>4</td></tr></table> <p>5</p>	.	5	..	2	a	9	b	4	<table><tr><td colspan="2">/tmp/c</td></tr><tr><td colspan="2"></td></tr></table> <p>6</p>	/tmp/c				<table><tr><td>.</td><td>6</td></tr><tr><td>..</td><td>2</td></tr><tr><td>files</td><td>8</td></tr><tr><td>data</td><td>4</td></tr></table> <p>7</p>	.	6	..	2	files	8	data	4
.	2																																										
..	2																																										
etc	6																																										
tmp	5																																										
c	4																																										
d	10																																										
/tmp/a																																											
.	5																																										
..	2																																										
a	9																																										
b	4																																										
/tmp/c																																											
.	6																																										
..	2																																										
files	8																																										
data	4																																										

Els blocs son de 4Kb. El superbloc es troba sempre a memoria i no hi ha cap tipus de buffercache. Si tenim el següent programa

```
int fd;
```

```
void thread1()
```

```
{
    char buf[100];
    int n,m;

    n = read(fd,buf,sizeof(buf));          2
    m = n;
    n = lseek(fd,SEEK_END,0);              3
    n = write(fd,buf,m);                   5
}
```

```
void thread2()
```

```
{
    char buf[100];
    int n;

    n = read(fd,buf,sizeof(buf));          4
    n = read(fd,buf,sizeof(buf));          6
}
```

```
int main ()
```

```
{
    fd = open("/etc/data",O_RDWR);  1
}
```

```

    create_thread(thread1);
    create_thread(thread2);
}

```

a) Calculeu quants accessos, indicant quins són, es fan al disc en cadascun dels punts indicats (suposant que es fan en l'ordre especificat). Quant val n en cada punt?

1. Accessos:

Tota d'accesos per aquesta instrucció = n =

2. Accessos:

Tota d'accesos per aquesta instrucció = n =

3. Accessos:

Tota d'accesos per aquesta instrucció = n =

4. Accessos:

Tota d'accesos per aquesta instrucció = n =

5. Accessos:

Tota d'accesos per aquesta instrucció = n =

6. Accessos:

Tota d'accesos per aquesta instrucció = n =

b) A quines estructures de dades s'accedeix al punt 4? Quines es modifiquen?

**22.-(Parcial 2005 QT)** Donat el següent sistema de fitxers amb inodes:

### Inodes

Tipus	dir	dir	link	dir	dat	dir	dat	link	dat
#links	4	3	2	1	1	1	1	1	1
BDs	1	2	3	6	4		7 3	5	
	2	3	4	5	6	7	8	9	10

### Blocs de dades

.	2	.	3	/A/D	/C/E	/A/B	.	3	abcd
..	2	..	2				..	2	
A	3	B	6				F	8	
B	4	D	4				E	9	
C	5	F	7				F	10	
1		2		3	4	5	6	7	

Trobeu quatre inconsistències, tot explicant perquè ho són.

- 1.
- 2.
- 3.
- 4.