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

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

bloque raíz

B	dir	3
H	dat	10
C	dir	2
D	dir	13
:	:	:

bloque 2

G	dat	14
P	?	11
R	dat	12
:	:	:

bloque 3

D	dat	4
E	dat	6
F	dat	7
:	:	:

bloque 11

/B/F

bloque 12

/B/E

bloque 13

b) 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?

c) De que tipo(s) puede ser el fichero *P*. Razónalo.

d) 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:

```
obrir(2, x, LECT, 0);
posicionar(r, y);
esperar(llegir(r, 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.- Examen final Q1 00-01. Donat el següent sistema de fitxers de MS-DOS, quants accessos a disc serien necessaris (especifiqueu quins) per obtenir el 4rt bloc de dades del fitxer /A/S? Considereu que la FAT no está 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

BD:

.	dir	2
..	dir	2
A	dir	7
B	dir	9

2

.	dir	7
..	dir	2
S	dat	8

7

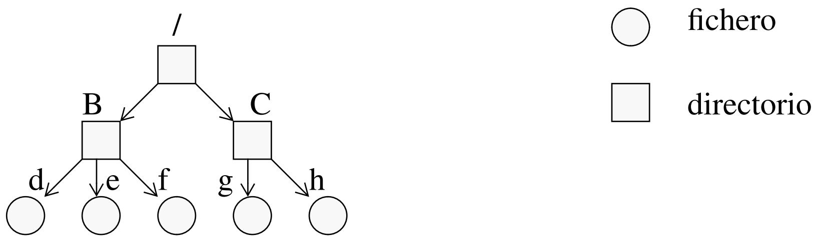
/B/R

8

.	dir	9
..	dir	2
R	dat	12
T	dir	15

9

3.- Tenemos un sistema de ficheros con un espacio de nombres en árbol, con los siguientes directorios, subdirectorios y ficheros:

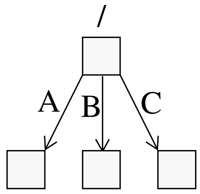


Si suponemos que:
- No tenemos estructuras de datos en memoria que nos ahorren accesos al disco. Sólo dispondremos de un buffer del tamaño de un sector.
- El tamaño de un registro es igual al de un sector, que es también la unidad de asignación de espacio.
- Se conoce el nº de bloque del dir / (FAT y sectores encadenados) o su nº de tabla de índices
- Los directorios ocupan un solo bloque.

Se pide:

- Indicar los accesos a disco necesarios para leer el bloque 10 del fichero *f*, suponiendo que el espacio se organiza con:
  - **a)** *Tabla de índices*. En la entrada al directorio de cada fichero está el número de bloque donde se encuentra su tabla de índices.
  - **b)** *FAT*. Ocupa un sector.
  - **c)** *Sectores encadenados*. No hay que considerar el tamaño de los punteros entre registros.
- Si el fichero *f* es un hard-link al fichero *g* , y suponiendo la organización de disco del caso a)
  - **d)** Dibujar cómo quedaría el sistema de ficheros.
  - **e)** Indicar los accesos a disco necesarios para leer el bloque 10 del fichero *f*.
- Si el fichero *f* es un soft-link al fichero *g* , y suponiendo la organización de disco del caso a)
  - **f)** Dibujar cómo quedaría el sistema de ficheros.
  - **g)** Indicar los accesos a disco necesarios para leer el bloque 10 del fichero *f*.

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)?

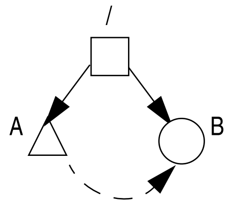
5.- En un sistema UNIX realizamos los siguientes comandos y el sistema devuelve el siguiente error:

$ ln -s Fich Link
$ rm Fich
$ cat Link
cat: cannot open Link: No such a file or directory
$

a) Por qué no se puede acceder al fichero Link si no ha sido borrado?
b) Indicar cómo se podrían modificar las estructuras de datos del sistema de ficheros de UNIX para que este error no se produzca y se pueda seguir accediendo al fichero Link. Indicar los campos que habría que añadir y en qué llamadas al sistema serían usados.
c) Sirve la solución que habéis planteado en b) si el *symbolic link* apunta a un fichero de otro disco?

6.-Examen final Q2 00-01. Es vol fer una modificació en la implementació del sistema de fitxers en lo referent als *soft-links*. Fins ara, un *soft-link* s´implementa mitjançant un i-node amb el tipus de fitxer *link* i un bloc de dades on s´indica el path del fitxer referenciat. Quins aventatges i inconvenients portaria implementar aquest mecanisme de manera que l´i-node del link contingués, per comptes de l´apuntador a un bloc de dades, el número de l´i-node del fitxer al que fa referència?

Exemple:



El fitxer A és un soft-link a B

Implementació actual

I-nodes

Fitxer	/	A	B
Tipus	dir	link	dat
1er BD	1	2	36
#i-node	2	3	4

Blocs de dades

#BD	1	2	36
	. 2 .. 2 A 3 B 4	/B	Soy el fichero B.

Implementació proposada

I-nodes

Fitxer	/	A	B
Tipus	dir	link	dat
1er BD o #i-node	1	4	36
#i-node	2	3	4

Blocs de dades

#BD	1	36
	. 2 .. 2 A 3 B 4	Soy el fichero B.

7.-Examen final Q1 02-03. 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?

8.-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
Tipo de fichero	dir	dir	dir	dir
Tabla de índice (1er bloque)	6	6	9	4

bloque 4

.	11
..	4
A	7
B	17

bloque 6

.	5
..	4
C	7
D	8

bloque 9

.	7
..	9
E	11
F	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"?

9.- Examen Final Q1 00-01. Donat el següent sistema de fitxers tipus Unix, indica les 3 inconsistències:

i-nodes

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*

Blocs de dades:

Bloc *6*

.	7
..	4
C	18
D	11

Bloc *9*: 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

Bloc *10*

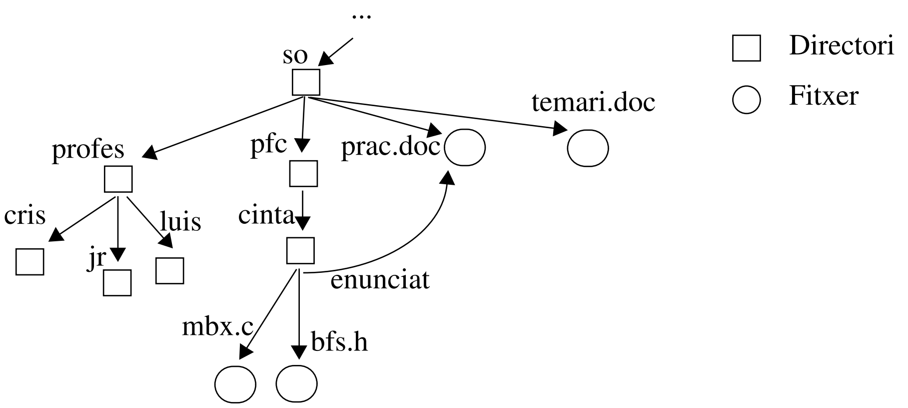
.	3
..	3
A	4
B	7
C	18

Bloc *15*: /A/B/D

Bloc *22*

.	4
..	3
B	7
C	18
L	21

10.- 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

Taula d' i_nodes

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:

Directori profes

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

bloc 60

Directori Cinta

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

11.- Examen Final Q1 01-02. Quin és l´error que presenta aquest sistema de fitxers DOS ?

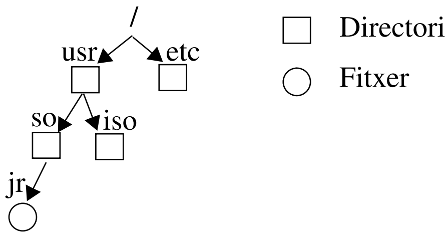
ROOT		
A	dir	1
B	dir	10

bloc 1		
x	dat	2
y	dat	5

bloc 10		
z	dat	11

FAT												
1	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

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

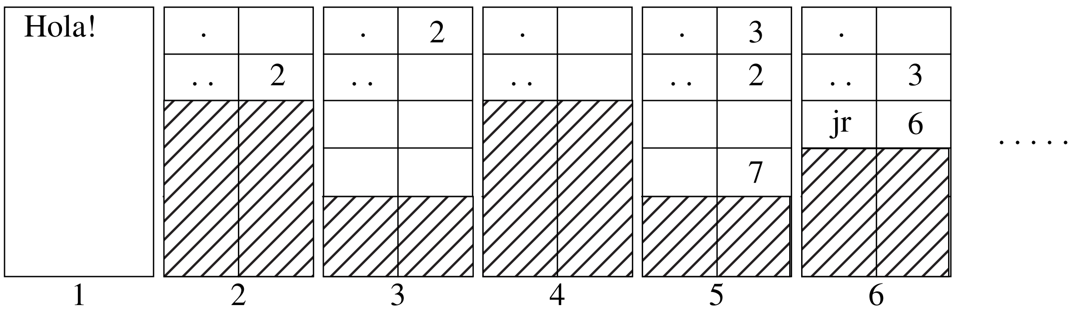


Sabent que l'organització emprada és la de UNIX, omplir els camps buits amb les dades que falten:

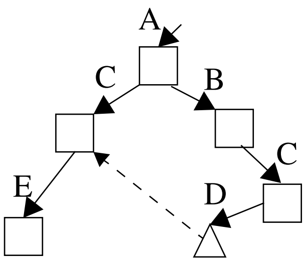
Taula d' i_nodes

Número d' i_node	2	3	4	5	6	7	.....
Tipus de fitxer				dir	data		
Índex al primer bloc		5	6	2	1	4	

Blocs de Dades



13.-



El fichero D es de tipo link. El contenido del fichero D es /A/C
En memoria tenemos el super-bloque.
En memoria tenemos espacio suficiente para mantener todo lo que se traiga de disco.

- Cuántos accesos a disco hacen falta para leer el bloque 15 de D?

a) Estructura de i-nodes. Cada i-node ocupa un bloque. El directorio A se encuentra en el i-node 2 (información del super-bloque).

b) FAT. La FAT ocupa un bloque. El directorio A se encuentra en el bloque 4 (información del super-bloque).

c) Observas algún problema en este sistema de ficheros?

14.- 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
Tipo	dir	link	dir	dat	link	dat
Índices	9	10	11	4 12 6 5	8	13 7

Bloques

8
/B

9	
.	2
..	2
A	7
B	5
F	9

10
/B/D

11	
.	5
..	2
C	4
D	6
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 <u>leer</u> 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 hacer 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^o 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 <u>leer</u> 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?

15.- Examen Final Q1 01-02. 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

16.- 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:

a) Dibuixar l'arbre de directoris corresponent a aquest sistema de fitxers, sabent que l'i-node 2 correspon al directori arrel.

b) 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 qué és el que llegiu en cada accés.

c) Dibuixeu com quedaria la FAT en un sistema de fitxers tipus DOS, suposant que els blocs de dades i els directoris estan al mateix lloc que en el cas de Unix.

d) Si esborrem el fitxer /B/D/r, indiqueu quins blocs de dades es veuen afectats en el cas de Unix i en el de DOS. En el cas de DOS, indiqueu tambe com quedaria la FAT.

17.- Examen Final Q2 99-00. 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 .. 2 A 4 fff 6	. 10 .. 4 fff 5	abc	. 8 .. 4 C 10 D 9 fff 7	. 9 .. 8 fff 3	ijk	rst	. 4 .. 2 B 8 E 10 fff 11	../fff	xyz

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, qué és el s'està llegint.

b) Quin és el contingut del fitxer llegit?

c) Quin és el nombre mínim de buffers (del tamany d´un bloc) que es necessita per a poder fer aquest accés. Raoneu la vostra resposta.

18.- Examen Final Q2 98-99. 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 *Directori*.
- 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.

19.- Control Aplicación Q2 00-01. 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:

Inodes:

2 (root)	**4**	**8**	**15**	**23**	**30**	
dir	dades	dir	dades	dir	link	*tipus*
...	...	...	...	...	...	
1	4	23	15	8	5	*1er bloc*
4	1	3	1	2	1	*#links*

Blocs de Dades:

1

.	2
..	2
A	8
B	23
f2	15

8

.	23
..	2
C	8

23

.	8
..	2
f1	30
f2	4

4

abc

5

/B/C/f2

15

xyz

a) -Assumint que disposem de un buffer a memòria <u>de tamany il.limitat</u> (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.

20.- Control Aplicación Q2 00-01. 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
..	2
A	15
E	4
C	9

3

with a sack on his back, And he put down his load where he thought

4

../D/F

5

.	9
..	15
M	11
F	14
L	19

6

/C/L

8

A long time ago came a man on a track, Walking thirty miles

10

.	15
..	2
G	11
R	4
T	16
D	9

11

/

13

it was the best, He made a home in the wilderness.

16

/C/L

26

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

b) Si actualmente estamos en el directoria 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);`

21.- La llamada al sistema de Onion

int **moure_fitxer** (nom_vell, nom_nou)
char *nom_vell, *nom_nou:

Cambia el nombre de un fichero de nom_vell a nom_nou. Sólo podrá ejecutarla su propietario ó el super-onion. La llamada devuelve los siguientes valores:

0: Correcto
-1: Fichero de nom_vell inexistente
-2: Ya existe un fichero con nom_nou
-3: Privilegios insuficientes.

Describid las estructuras de datos (indicando los campos), tanto a **nivel sistema** como a **nivel bfs**, que se ven afectadas (consulta ó modificación).
Hacer un esquema donde se vean qué procesos ó rutinas actúan sobre estas estructuras de datos.

22.- Para ejecutar la llamada al sistema de Onion:

int **esborrar_fitxer** (nom_fitxer)
char *nom_fitxer

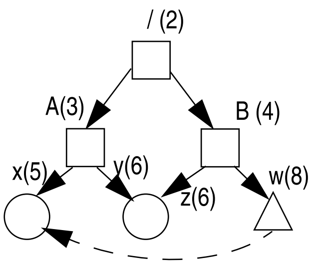
a) Describid qué estructuras de datos (especificando los campos pertinentes), tendríamos que actualizar ó consultar, tanto a **nivel sistema** como a **nivel bfs**.
Si queremos modificar esta llamada para permitir, en lo posible, recuperar un fichero que ha sido borrado

b) Añadirías alguna nueva llamada a sistema? En tal caso, indicar sus parámetros.
c) Explicar el comportamiento de las llamadas a sistema implicadas.
d) A partir de qué momento deja de poder recuperarse un fichero?

23.- Examen Final Q1 00-01. Considerant les estructures de dades a memoria d'un sistema de fitxers d'ONION, quines d'aquestes s'accedeixen o es modifiquen quan es fa la crida a sistema **posicionar**(fd,3*TAM_BD) si considerem que TAM_BD és el tamany del bloc de dades.

24.- Examen Final Q2 01-02. ¿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?

25.- Examen Final Q1 02-03. Suposeu que teniu el següent sistema de fitxers de la figura de tipus Unix (entre paréntesis, el número d´i-node). Dibuixeu com queden les Taules de Canals, els Descriptors de Dispositiu i la Taula de Fitxers Estàtica (suposant que Onion treballa amb i-nodes) si s´executen les següents operacions. Nota: Considereu que inicialment no hi ha cap altre procés treballant amb cap fitxer i crear-pcb provoca l´herència dels 3 primers canals..

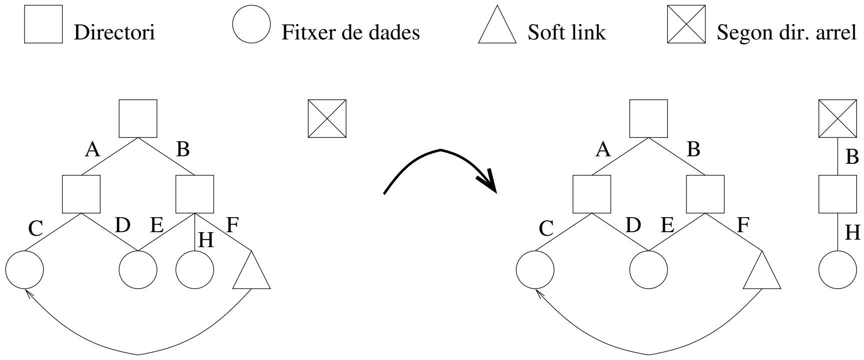


Proces 1: obrir(1,"/A/x", Lect);
obrir(4,"/A/y", Esc);
crear_pcb(Proces3,1,2);

Proces 2: obrir(3, "/B/z", Lect);

Proces 3: obrir(4, ¨/B/w", Lect);

26.- Examen Final Q2 00-01. Es proposa un sistema de fitxers basat en el de UNIX, però que permet recuperar els fitxers esborrats. Per tal d'implementar aquesta recuperació, es crea un segon arbre de directoris invisible a l'usuari que conté els fitxers esborrats i tot l'arbre de directoris que sigui necessari per tal de poder recuperar els fitxers en el seu lloc d'origen. Quan cal espai lliure al disc, es recicla l'espai ocupat pel fitxer esborrat més antic.



D'aquesta manera, i tal i com es veu a l'exemple, quan esborrem un fitxer (rm /B/H) el que es fa realment és traslladar el fitxer al segon arbre de directoris (com si fèssim un *move*), tot creant els subdirectoris que siguin necessaris. Si volem recuperar un fitxer, només caldrà fer l'operació inversa: fer un *move* del fitxer des del segon arbre de directoris cap al seu lloc original. Si després de recuperar un fitxer algun directori del segon arbre ens queda buit, reciclarem l'espai ocupat per aquell directori (és a dir, l'esborrarem de forma permanent).

Es demana que respongueu a les següents qüestions:

a) Doneu números als inodes i blocs de dades que necessiteu, i dibuixeu la taula d'inodes i els blocs de dades dels directoris per tal de reflectir els arbres de directoris de l'exemple (després d'esborrar /B/H).

b) Quan esborrem un fitxer que té més d'un *link*, tenim dues possibilitats: fer un *move* tal i com hem fet a l'exemple, o fer-ne una còpia (és a dir, amb el *move* continuem apuntant als mateixos blocs de dades i amb la opció de *còpia* fem una duplicació dels blocs de dades al segon directori) . Si ara esborrem el fitxer /B/E, per cada una de les dues opcions (*move* i *copy*) digueu:
- Com queda afectat el nombre de *links* de /A/D?
- Qué passa si modifiquem /A/D, i després recuperem el fixter /B/E?

c) Sabent que tots els inodes estan guardats al mateix bloc de disc (és a dir, que tots els inodes ocupen un únic bloc de dades), indiqueu quins accessos a disc serien necessaris per fer un *undelete* /B/H després d'haver esborrat /B/E. Suposeu que els dos inodes arrel es troben indicats en el superblock, que el superblock sempre està a memòria, i que no tenim cap altre informació a memòria.
- Si sols tenim un *buffer* d'un bloc.
- Si disposem d'una *buffer cache* de 4 blocs amb substitució LRU

d) Descriu breument com implementariem la mateixa idea en un sistema de fitxers amb FAT en el que es permet tenir *links* de qualsevol tipus.

27.- Examen Final Q2 01-02. Es disposa de la següent llista d´i-nodes i blocs de dades en un sistema de fitxers basat en taula d´indexos:

I-nodes:

	dir	link	dir	dat	link	dir	dat	dat
^BDs	2	7	6	13,9,20	4	18	8,21	19,15
#i-node	*2*	*6*	*8*	*13*	*18*	*19*	*20*	*23*

Blocs de Dades

#BD *2*:

.	2
..	2
A	8
B	18

#BD *4*:

/A/C/B/B

#BD *6*:

.	8
..	2
A	23
B	19
C	6
D	20

#BD *7*:

/A

#BD *18*:

.	19
..	8
A	20
B	18

I dels següents processos (entre corxets apareix el temps que triga en executar-se cada part i un guió significa que el seu cost en temps és 0). Considereu que l =TAMANY_BLOC:

```
void P1() {
 init_proces1();              [3]
 crear_pcb(P2,3,3);           [-]
 preparar_p3();               [1,8]
 crear_pcb(P3,2,2);           [-]
 procesar();                  [0,2]
 obrir(3,"/A/C/D","r");       [X]
 procesar();                  [4]
 esperar(llegir(3,b,l));      [Y]
 procesar();                  [1,5]
 esperar(llegir(3,b,l));      [Z]
 tancar_i_destruir_P1();[1]
}
```

```
void P2() {
 init_proces2();       [2,5]
 retardar(4);          [4]
 procesar();           [5]
 retardar(4);          [4]
 destruir_P2();        [2]
}
```

```
void P3() {
 init_proces3();              [2]
 retardar(4);                 [4]
 procesar();                  [4]
 obrir(3,"/A/A","r");         [W]
 procesar();                  [3]
 esperar(llegir(3,b,l));      [T]
 tancar_i_destruir_P3();  [1]
}
```

a) Dibuixa com quedaria la FAT d´aquest sistema de fitxers suposant que els blocs de dades i els directoris ocupen el mateix lloc que al sistema original i que no hi ha més blocs de dades ocupats.

b) . Si considereu que el cost en unitats de temps de les crides a sistema sobre fitxers es pot calcular mitjançant la fórmula **1+0,5*#accessos_a_disc**, calculeu quin seria el valor de X,Y,Z,W i T si:

- El superbloc sempre està carregat a memòria.
- Disposem d´una buffer cache de tamany 1 bloc.
- El tamany dels i-nodes i dels blocs de dades és d´1 bloc.

- Totes les crides a sistema de llegir fan la lectura d´un bloc sencer.
- Disposem dels TADs de memòria del sistema de fitxers (taula d´i-nodes)

c) Dibuixeu el diagrama de Gantt corresponent a l´execució concurrent d´aquests processos si assumiu que a l´instant t=0 comença a executar-se el procés P1 i que la política de planificació és Round Robin amb prioritats apropiatives immediates. Considereu que el procés P1 té prioritat 2 i quantum 2 i que durant X, Y, Z, W i T el procés implicat està bloquejat tot aquell temps.

d) Calcula el temps mig d´espera i de retorn dels 3 processos per la política anterior.

e) Només pel cas de les crides a sistema d´obrir, recalculeu el cost d´aquestes crides si disposem d´una buffer cache amb 6 entrades i política de reemplaç LRU. D´igual manera, mostreu el contingut de la buffer cache després de cada accés. Nota: L´ordre de les operacions és: primer l´obrir del procés P1 i després l´obrir del procés P3 i no hi ha cap més accés a la buffer cache entre mig d´aquestes dues operacions.

28.- Examen Final Q1 02-03. 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	
								?	0	
#BD	*2*		*6*	*9*		*8*		*13*		*18*

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

b) Quins accesos cal fer a disc, i quin és el graf d'aquest sistema de fitxers després de fer els accesos 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 accesos, 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).

c) Tenim un sistema amb Round Robin amb quantum de 100 milisegons i els processos A,B,C arriban al sistema simultàniament i en aquest ordre, quina és la seqüència d'execució sobre **el sistema de fitxers de l'apartat a)**. Assumiu que el cost de les operacions és despreciable excepte els accessos a disc on primer s´ha de programar el controlador (cost 10ms de CPU) i que les dades són disponibles 50ms després. Assumiu també que el superbloc sempre és a memòria, que disposeu d´una buffer cache de tamany infinit, inicialment buida i que poden haver múltiples peticions de disc simultàniament sense que afecti al seu cost. Les operacions d´E/S write i read **una vegada està el bloc a la buffer cache** (en ambdos cassos) tenen un cost de 10ms.

Nota: La rutina de multiplexar entra 200 vegades cada segon.

```
A:     fd = open("/c/h", O_WRONLY|O_CREAT, 0600);
       write (fd, "A",1);
       close (fd);
B:     fd = open("/a/f", O_WRONLY|O_CREAT, 0600);
       write (fd, "B",1);
       close (fd);
C:     fd = open("/c/h", O_RDONLY);
       read (fd, &c,1);
       close (fd);
```

Què conté el bloc de dades 18 després de l´execució? I la variable c?

29.- Examen Final Q2 02-03. 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	dir	link	dir	dat	link
^BDs	3,8	12	13	4	9	144	32,16	14
#i-node	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

Blocs de Dades

#BD	*3*		*4*		*8*		*9*	*12*		*13*	
	.	2	.	5	G	9	../../G	.	3	.	4
	..	2	..	3	B	0		..	2	..	2
	A	3			G	0		C	5	E	7
	B	4			A	0		D	6	F	8

#BD	*14*	*16*	*32*	*144*	
	/B/F		/A/B/F	.	7
				..	4

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

b) Què passaria si tenim els següents blocs de dades i i-nodes al sistema de fitxers anterior?Considereu-los cassos independents.

	dir
^BDs	12,8
#i-node	*3*

G	9
B	0
G	4
A	0
8	

/A/D
9

c) Calcular el número d'accessos a disc necessaris per realitzar les següents seqüències de crides al sistema, tenint en compte que no tenim res a la buffer cache (ni el superbloc), que la buffer cache té 2 entrades i algorisme LRU. Els i-nodes ocupen 1 bloc. Indiqueu la seqüència d'accessos, tot assenyalant si són de lectura o d'escriptura. Per indicar-ho, 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 encerclant-les.

c.1)
```
fd = open("/A/D", O_RDWR|O_TRUNC|O_CREAT, 0666);
write (fd,"a",1);
close (fd);
```
c.2)
```
fd = open("/A/D", O_RDWR);
write (fd,"a",1);
close (fd);
```