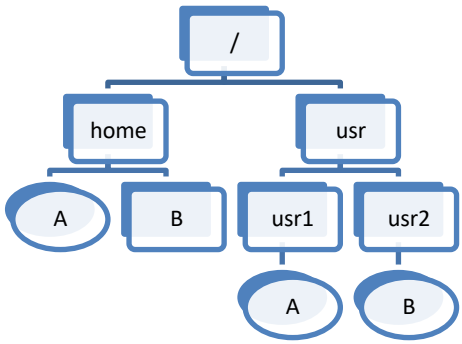
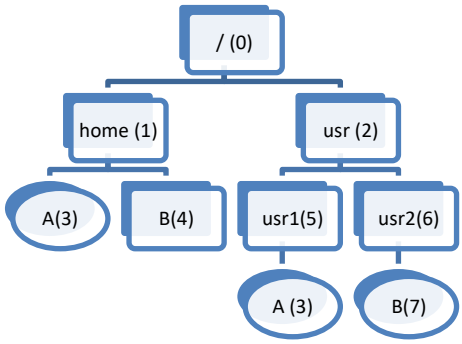
Ejercicio 8 (T4)

Tenemos el siguiente esquema de directorios con la siguiente información adicional:



- Sabemos que el tamaño de un bloque de disco son 512 bytes y que un inodo ocupa 1 bloque.
- Sabemos que /home/A y /usr/usr1/A son hard-links. El fichero ocupa 2 KB.
- Sabemos que /usr/usr2/B es un soft-link a /home/usr/usr3 (un directorio que ya no existe)
- Los ficheros marcados con cuadros son directorios
- El kernel implementa la optimización de buffer cache, compartida para inodos y bloques de datos.

1. En el siguiente dibujo, asigna números de inodos a todos los ficheros y directorios. Ten en cuenta el tipo de fichero a la hora de hacerlo.



2. Completa el siguiente dibujo añadiendo la información de los inodos y el contenido de los bloques de datos que intervienen. Utiliza las mismas etiquetas que Linux para indicar el tipo de fichero (d=directorio,-=fichero datos, l=soft-link). (por simplicidad hemos puesto espacio solo para 5 bloques de datos). La asignación de bloques de datos se hace de forma consecutiva.

TABLA DE INODOS EN DISCO

Num_inodo	0	1	2	3	4	5	6	7	8	9	10
#enlaces	4	3	4	2	2	2	2	1			
Tipo_fichero	d	d	d	-	d	d	d	L			
Tabla de indices	0	1	2	3	7	8	9	--			
				4							
				5							
				6							

Bloques datos

Num Bloque Datos	0	1	2	3	4	5	6	7	8
	. 0 .. 0 home 1 usr 2	. 1 .. 0 A 3 B 4	. 2 .. 0 usr1 5 usr2 6	AAA	AAAA	AAA	AAA	. 4 .. 1	. 5 .. 2 A 3

Num Bloque Datos	9	10	11	12	13	14	15	16	17
	. 6 .. 2 B 7								

NOTA: En esta solución se ha supuesto que el path al que apunta /usr/usr2/B cabe en su inodo y no necesita un bloque diferente

3. Si nuestro directorio actual es /home:
   a) ¿Cuál es el path absoluto para referenciar el fichero /usr/usr1/A?

/usr/usr1/A

   b) ¿Cuál es el path relativo para referenciar el fichero /usr/usr1/A?

../usr/usr1/A

4. Dado el siguiente código, ejecutado en el sistema de ficheros de la figura, contesta (justifícalas todas brevemente):

```
1. char c;
2. int fd,fd1,s,pos=0;
3. fd=open("/usr/usr1/A",O_RDONLY);
4. fd1=open("/usr/usr1/A",O_WRONLY);
5. s=lseek(fd1,0,SEEK_END);
6. while(pos!=s){
7.     read(fd,&c,sizeof(char));
8.     write(fd1,&c,sizeof(char));
9.     pos++;
10. }
```

a) Describe brevemente qué hace este código

Este programa duplica el contenido de un fichero en el mismo fichero. Nos ponemos al final y vamos leyendo (del inicio) y escribiendo(al final). Controlamos el final de la operación con el tamaño del fichero.

b) Indica exactamente qué inodos y bloques de datos accede la línea 3.

```
fd=open("/usr/usr1/A",O_RDONLY);
```

Leemos el inodo de /(0) , el bloque de datos de /(0). El inodo de usr(2), el bloque de datos de usr(2). El inodo de usr1(5) + el bloque de datos de usr1(8). El inodo de A(3). TOTAL 7 accesos

c) Asumiendo que ninguna llamada a sistema da error, ¿cuántas iteraciones dará el bucle de las líneas 6-10?

El fichero tiene tamaño 2kb, por lo tanto dará 2x1024=2048 iteraciones

d) ¿Qué bloques de datos accede la línea 7?

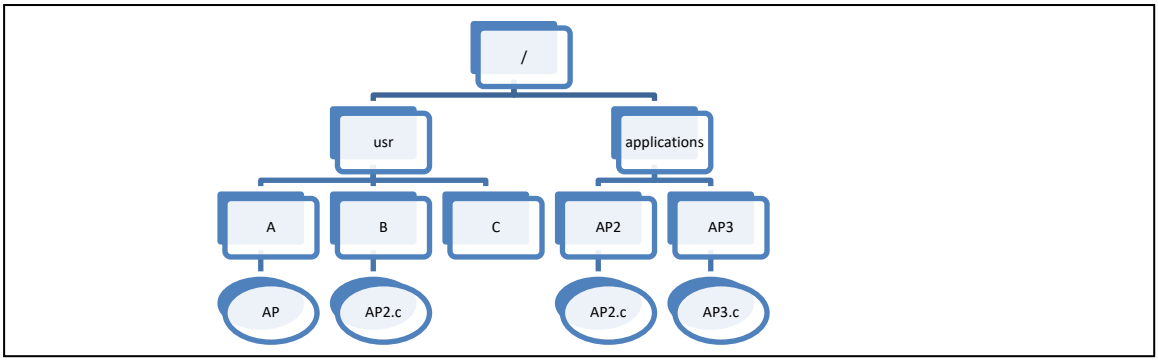
Los bloques de disco son de 512 bytes, para leer los 2K serán necesarios 4 bloques.

e) ¿Qué valor tendrá el puntero de lectura/escritura del canal indicado en fd al finalizar el bucle?

El puntero valdrá 2x1024=2048.

Ejercicio 11 (T4)

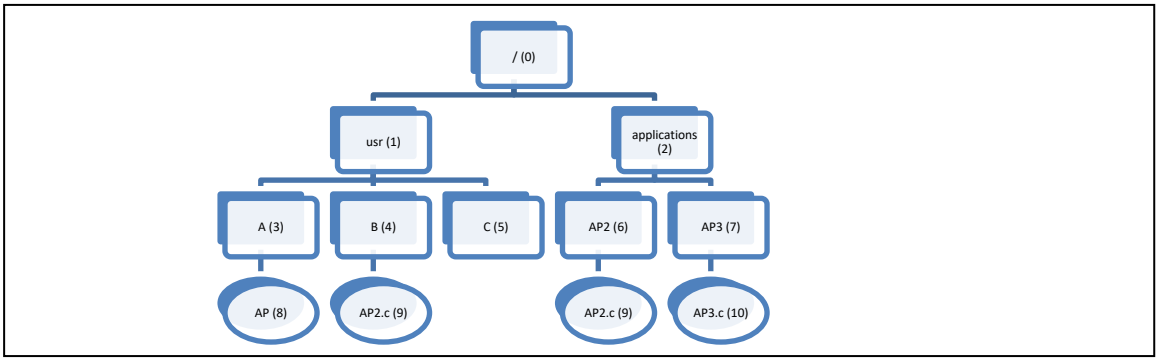
Dado el siguiente esquema de directorios en un sistema de ficheros basado en inodos:



Del cual nos dicen que /usr/A/AP es un soft-link a /applications/AP2/AP2.c y que /usr/B/AP2.c y /applications/AP2/AP2.c son hard-links. Sabiendo que:

- tipo de datos puede ser dir = directorio, dat= fichero normal, link=softlink.
- El directorio raíz es el inodo 0.
- Los cuadrados indican directorios.
- El fichero AP3.c ocupa 3 bloques de datos y el resto 1 bloque de datos.

1. Rellena los siguientes inodos y bloques de datos. Para ello:
   a) Asigna primero un inodo a cada elemento del esquema de ficheros y directorios y anótalo junto a su nombre en la figura.



b) Asigna bloques de datos a cada elemento del esquema de directorio teniendo en cuenta los tamaños que os hemos indicado.
c) Completa la figura de la tabla de inodos rellenando los campos que aparecen y los bloques de datos con la información que conozcas sobre su contenido

Tabla de Inodos en disco

Num_inodo	0	1	2	3	4	5	6	7	8	9	10
#links	4	5	4	2	2	2	2	2	1	2	1
Tipo_fichero	dir	Dir	dir	Dir	dir	Dir	dir	dir	Link	Data	Data
Tabla de indices	0	1	2	3	4	5	6	7	-	8	9,10,11

Bloques datos

Num Bloque	0	1	2	3	4	5	6	7
Contenido	. 0 .. 0 Usr 1 Applications 2	. 1 .. 0 A 3 B 4 C 5	. 2 .. 0 AP2 6 AP3 7	. 3 .. 1 AP 8	. 4 .. 1 AP2.c 9	. 5 .. 1	. 6 .. 2 AP2.c 9	. 7 .. 2 AP3.c 10

Num Bloque	8	9	10	11	12	13
Contenido	Cont. AP2.c	Cont .AP3.c	Cont. AP3.c	Cont. AP3.c		

Justificación:

- Cada fichero o directorio tiene un inodo, excepto los hard-links que apuntan al mismo inodo, por eso no necesitamos 11 sino 10.
- Los soft-links son ficheros especiales con el path del fichero a que apuntan. En este ejercicio se asume que el path cabe en el propio inodo y no necesita un bloque extra.
- El contenido de un directorio es una tabla con dos columnas: nombre fichero y número de inodo del fichero. Un directorio siempre incluye dos ficheros de tipo directorio: el . y ..
- El fichero . hace referencia al inodo del directorio en el que se está en ese momento y el .. al inodo del directorio padre
- Las referencias de un inodo es el número total de nombres de ficheros que apuntan a ese inodo (incluidos . y ..)
- Bloques de datos es la lista de bloques de datos de un fichero. Si hubieran mas de 10

2. Indica cuántos accesos a bloques e inodos (y cuáles) hacen las siguientes llamadas a sistema (indícalas una por una). Supón que, en el momento de ejecutar esta secuencia de llamadas a sistema, el sistema acababa de iniciarse y ningún otro proceso estaba usando ningún fichero del sistema de ficheros. Sabemos que un bloque son 4096 bytes.

Aclaración:

- El open accede a los inodos necesarios hasta llegar al inodo destino, pero no lee ningún bloque de datos del fichero destino
- Al hacer un open con el flag O_CREAT, hay que crear un fichero nuevo (ya que no existe). Eso implica: un inodo nuevo, escribir el bloque de datos del directorio para añadir un fichero, cambiar el tamaño del directorio para indicar que ha variado su tamaño.

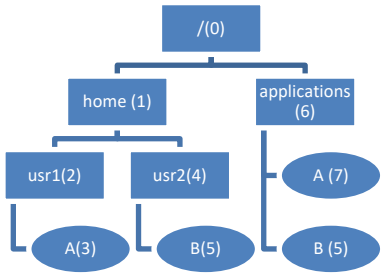
```
char buff[4096];

1. fd=open("/applications/AP3/AP3.c",O_RDONLY);
2. fd1=open("/usr/C/AP3.c",O_WRONLY|O_CREAT,0660);
3. ret= read(fd,buff,sizeof(buff));
4. lseek(fd,4096, SEEK_CUR);
5. write(fd1,buff,ret);
6. close(fd1);
```

	Accesos a bloques e inodos (cuáles)	Tablas Modificadas (SI/NO)		
		Canales	F. Abiertos	Inodos
1	I0+B0+I2+B2+I7+B7+I10	SI	SI	SI
2	I0+B0+I1+B1+I5+B5+Nuevo inodo (11?)	SI	SI	SI
3	B10, primero de AP3.c	NO	SI	NO
4	Ninguno.	NO	SI	NO
5	Nuevo Bloque (B13?), para fichero nuevo.	NO	SI	SI
6	I11+(B5+I5), por el O_CREAT	SI	SI	SI
7	I10	SI	SI	SI

Ejercicio 15 (T4)

Supón un sistema de ficheros basado en inodos que tiene la siguiente estructura (los cuadrados indican directorios, entre paréntesis hemos puesto el número de inodo):



Sabemos que:

- /home/usr1/A es un soft-link a /applications/A
- /home/usr2/B es un hard-link a /applications/B
- /applications/B es un fichero vacío
- /applications/A ocupa 1000 bytes
- Cada directorio y cada inodo ocupan 1 bloque de disco
- El tamaño de un bloque de disco es de 512bytes

Y ejecutamos la siguiente secuencia de código:

```
1. char c[100];int i,fd_in,fd_out,ret;
2. fd_in=open("/home/usr1/A",O_RDONLY);
3. fd_out=open("/home/usr2/B",O_WRONLY);
4. ret=read(fd_in,c,sizeof(c));
5. while(ret>0){
6.    for(i=0;i<ret;i++) write(fd_out,&c[i],1);
7.    ret=read(fd_in,c,sizeof(c));
8. }
```

Contesta las siguientes pregunta:

1. ¿Qué inodo(s) cargaremos en la tabla de inodos en memoria al ejecutar la llamada a sistema de la línea 2?

Cargaremos el inodo 7 que es al que apunta /home/usr1/A →/applications/A. En la tabla de inodos en memoria solo se guarda el inodo destino, no los intermedios.

2. ¿Cuántos y cuáles accesos a inodos y bloques deberemos hacer para ejecutar la llamada a sistema de la línea 2? Indica cuáles corresponden a inodos y cuáles a bloques de datos (aunque no pongas el número de los bloques de datos).

Debemos acceder primero /home/usr1/A y luego a /applications/A

Inodo /+Datos dir /+ inodo home + Datos dir. Home + inodo usr1 + BD dir usr1 + inodo A + inodo / + BD dir / + inodos applications + BD applications dir +inodo 7= 13 accesos

Nota: se asume que el path /applications/A cabe en el inodo del soft link y no requiere un bloque separado

3. Si sabemos que una llamada a sistema tarda 10ms, ¿Cuánto tiempo invertirá este código (EN TOTAL) sólo en llamadas a sistema? (Indica el coste en ms al lado de cada línea de código y el total al final).

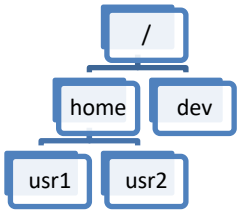
ACLARACIÓN: El fichero tiene 1000 bytes, por lo que se hacen 10 lecturas (sizeof(c) =100) y una última llamada a sistema que devuelve 0. Las escrituras se hacen byte a byte, por lo que hay 1000

```
1.  char c[100];
2.  fd_in=open("/home/usr1/A",O_RDONLY); ➔10ms
3.  fd_out=open("/home/usr2/B",O_WRONLY);➔10ms
4.  ret=read(fd_in,c,sizeof(c)); ➔10ms
5.  while(ret>0){
6.    for(i=0;i<ret;i++) write(fd_out,&c[i],1); ➔10x100=1000ms
7.    ret=read(fd_in,c,sizeof(c));➔10ms
8.  }

TOTAL= 10+10+10+(1000+10)*10=30+10100=10130ms
```

Ejercicio 19 (T4)

Tenemos el siguiente sistema de ficheros (todo son directorios):



Y nos dan el siguiente código:

```
1. void f(int i){
2.   char c,buff[64];
3.   int fd;
4.   sprintf(buff,"/home/usr1/%d",i);
5.   fd=open(buff,O_WRONLY|O_CREAT,0777);
6.   while(read(0,&c,1)>0) write(fd,&c,1);
7. }
8. void main(int argv,char *argv[]){
9.    int i,pid;
10.     for(i=0;i<4;i++){
11.      pid=fork();
12.      if(pid==0){
13.       f(i);
14.       exit(0);
15.      }
16.     }
17.     f(i);
18.     while(waitpid(-1,null,0)>0);
19. }
```

1. Describe brevemente que hace este código (procesos que se crean, que hace cada uno , etc)

Este código crea 4 procesos, todos ejecutan la función f que lee de la entrada std y guarda lo que lee en un fichero nuevo para cada proceso. El padre también ejecuta la función y al acabar espera a los hijos.

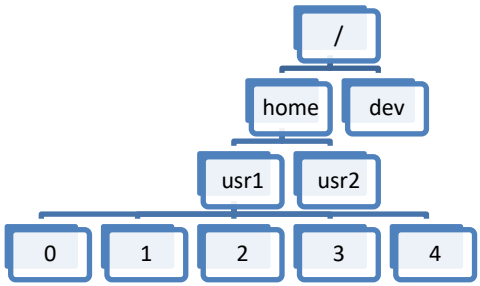
2. Sabiendo que, al inicio del programa, solo están creados los canales 0,1 y 2, ¿qué valor tendrá fd para cada uno de los procesos?

Para todos valdrá 3

3. ¿Cuántas entradas en la Tabla de ficheros abiertos generará este código?

5, ya que hay 5 open's en total

4. Dibuja como quedará el sistema de ficheros al acabar el programa



5. Sabemos que la entrada std es la consola, y sabemos que el usuario teclea los siguientes caracteres *1234567890* y a continuación apreta ctr-D, ¿Cuál será el contenido de cada fichero que se genera?

No lo podemos saber pq se ejecutan de forma concurrente

6. ¿Cuántos accesos a bloques de datos e inodos se producirán en total como consecuencia del bucle de la línea 6?

El read, al ser de la consola, no generará accesos a disco, el write, al ser byte a byte y no tener buffer cache, generará tantos accesos como write's se realicen, es decir 10 (para '*1234567890*').

7. Antes antes de ejecutar este código tenemos los inodos y bloques de datos con el siguiente contenido. Modifícalo para representar como quedará para este caso concreto: asume que los procesos se ejecutan en orden de creación (i=0,1,2,3,4) y que cada proceso lee una letra de la entrada std.

TABLA DE INODOS EN DISCO

Num_inodo	0	1	2	3	4	5	6	7	8	9	10
#enlaces	4	4	2	2	2						
Tipo_fichero	d	d	d	d	d						
Tabla de índices	0	1	2	3	4						

Bloques datos

Num Bloque Datos	0	1	2	3	4	5	6	7	8
	. 0 .. 0 home 1 dev 2	. 1 .. 0 usr1 3 usr2 4	. 2 .. 0	. 3 .. 1	. 4 .. 1				

Num Bloque Datos	9	10	11	12	13	14	15	16	17