Sistemas Operativos

Segundo Parcial. Sistema de Ficheros. 18/11/2020

Problema 1

Sea el siguiente programa en ejecución por el proceso P, del cual se han obviado algunos fragmentos denotados por puntos suspensivos, así como el control de errores, por mayor simplicidad.

```
int main(int argc, char **argv)
{
    char buf[1024];
    int fd1, fd2, n;

    /* ... */

    /* Instante inicial */
    n = write(1, "abcd", 4);
    n = read(4, buf, 1);

    switch (fork())
    {
    case -1:
        exit(1);
    case 0:
        fd1 = open("f", O_WRONLY | O_CREAT, 0666);
        lseek(fd1, 4, SEEK_CUR);
        n = read(4, buf, 5);
        n = write(fd1, buf, n);

        /* Apartado B */
        /* Apartado D */
    default:
        fd2 = creat("f", 0666);
        n = write(fd2, buf, 5);
        n = read(4, buf, 3);
        n = write(fd2, buf, n);
        lseek(fd2, 16, SEEK_CUR);

        /* Apartado A */

        /* ... */
    }

    /* ... */

    return 0;
}
```

En un instante inicial, la ejecución del programa ha avanzado hasta el punto marcado en el código mediante el comentario `Instante inicial`, encontrándose las tablas auxiliares del sistema operativo en el siguiente estado:

Tabla de procesos

BCP (P)	
UID: 100	
GID: 100	
fd:	
0	teclado
1	monitor
2	monitor
3	0
4	1
5	0

BCP ( )	
UID:	
GID:	
fd:	
0	
1	
2	
3	
4	
5	

Tabla intermedia

	i-nodo	**Posición**	**nrefs**	**Modo R/W**
1	678	7	1	10
2				
3				
4				
5				

Tabla de copias de i-nodos

i-nodo	**nopens**	**Agrup.**	**Tamaño**
678	1	134	13

A partir de este punto, se reanuda la ejecución del programa. Conteste a las siguientes preguntas teniendo en cuenta las siguientes consideraciones:

- El sistema de ficheros tiene un tamaño de agrupación de 4 KiB, y tiene libres todas las agrupaciones desde la número 1012, y todos los i-nodos desde el número 748, ambos incluidos. Agrupaciones e i-nodos se asignan en orden secuencial ascendente.
- El directorio de trabajo no tiene ningún fichero ni directorio denominado “f”.
- El usuario con UID 100 tiene permisos de escritura y acceso sobre el directorio de trabajo.
- La agrupación 134 contiene la siguiente información en el instante inicial:

```
0123456789ABCemplo de los efect…
```
(continúa)

- La variable `buf` tiene el siguiente contenido en el instante inicial:

```
abcdefghijklmnopq…
```
(continúa)

a) **[0.75 puntos]** Denomine H al proceso hijo que crea el proceso P tras la llamada al servicio `fork`. Suponga que el proceso P continúa su ejecución desde el instante inicial hasta el comentario marcado como `Apartado A`, quedando el proceso H en estado de espera sin haber comenzado su ejecución. Dibuje el estado de las tablas del sistema operativo en dicho instante.

Tabla de procesos

BCP (P)	**BCP (H)**
UID:	**UID:**
GID:	**GID:**
fd:	**fd:**

0	
1	
2	
3	
4	
5	

0	
1	
2	
3	
4	
5	

Tabla intermedia

	i-nodo	**Posición**	**nrefs**	**Modo R/W**
1				
2				
3				
4				
5				

Tabla de copias de i-nodos

i-nodo	**nopens**	**Agrup.**	**Tamaño**

b) **[0.75 puntos]** Tras el instante del apartado anterior, el sistema operativo pone en espera al proceso P y pone a ejecutar al proceso H. Vuelva a dibujar las tablas del sistema operativo de acuerdo al estado en que se encontrarían en el instante en que el proceso H alcanzase la marca con el comentario `Apartado B`.

Tabla de procesos

BCP (P)	**BCP (H)**
UID:	**UID:**
GID:	**GID:**
fd:	**fd:**

0	
1	
2	
3	
4	
5	

Tabla intermedia

	i-nodo	**Posición**	**nrefs**	**Modo R/W**
1				
2				
3				
4				
5				

Tabla de copias de i-nodos

i-nodo	**nopens**	**Agrup.**	**Tamaño**

c) **[0.5 puntos]** ¿Cuál sería el contenido del fichero “f” al término del apartado anterior?

d) **[1 punto]** Suponga que existe un fichero ejecutable en el directorio de trabajo denominado “prog” que lee datos por entrada estándar y devuelve el resultado por salida estándar. Escriba las líneas de código necesarias en la marca `Apartado D` para que el proceso H ejecute “prog” (empleando el servicio `exec`), de modo que su entrada sea el contenido del fichero abierto en el descriptor número 4, desde el inicio del fichero, y su salida se vuelque en el fichero abierto en el descriptor almacenado en la variable `fd1`.

SOLUCIÓN:

a)

En primer lugar, el proceso P escribe 4 bytes en el descriptor 1, es decir, por salida estándar. En base al contenido de la tabla de procesos, esos 4 bytes se mostrarán por pantalla, no afectando por tanto al estado de las tablas. Posteriormente, se lee 1 byte desde el descriptor 4, el cual apunta a la entrada 1 de la tabla intermedia, que corresponde al fichero con i-nodo 678 abierto en modo sólo lectura. Puesto que su puntero se encuentra en la posición 7, se leería el octavo byte del fichero, que es el carácter "7". El puntero pasaría a valer 8. El contenido de `buf` pasaría a ser `7bcdefghijklmnopq`….

Tras la ejecución del servicio `fork`, se crea el proceso H copiando los campos UID, GID y la tabla de descriptores de fichero del proceso P. Como consecuencia, el campo nrefs de la entrada 1 de la tabla intermedia se incrementa en 1. Es importante tener en cuenta, por tanto, que tanto el proceso P como el H compartirán el mismo descriptor.

El proceso P continúa ejecutando el código correspondiente a la rama `default` del bloque condicional, y llama al servicio `creat`. El resultado de esta llamada es la creación (puesto que no existe) y apertura del fichero "f", al cual se le asigna el i-nodo 748 (el primero libre). Inicialmente, no tendría agrupaciones asignadas y su tamaño sería 0. Se crea también la entrada número 2 en la tabla intermedia apuntando al i-nodo 748, inicializando nrefs a 1, la posición del puntero a 0 y el modo de apertura a sólo escritura. La entrada escogida de la tabla de descriptores para almacenar este nuevo descriptor del proceso P sería la número 3 (la primera libre) con el valor 2.

A continuación, se escriben 5 bytes en el fichero "f" desde `buf`, para lo cuál será necesario asignarle una agrupación (la primera libre, es decir, la 1012). El contenido del fichero pasa a ser `7bcde`, y su tamaño 5 bytes. Posteriormente, se leen 3 bytes del descriptor número 4. Recordemos que su puntero se encuentra en la posición 8, por lo que leerá 3 bytes desde dicha posición. Así, el contenido de `buf` será `89Adefghijklmnopq`…, y el puntero de la entrada 1 de la tabla intermedia pasa a valer 11. Esos 3 bytes recién leídos se escriben en el fichero "f", lo que actualiza su contenido a `7bcde89A`, su puntero al valor 8 y el tamaño almacenado en el i-nodo a 8. Finalmente, la llamada `lseek` modifica el valor del puntero de la entrada apuntada por el descriptor `fd2`, es decir, la 2, sumándole 16 debido al flag `SEEK_CUR`. Como resultado, el puntero pasará a valer 24. Nótese que esto no altera el tamaño del fichero, puesto que el agujero no se haría efectivo hasta realizar una escritura más allá del límite del fichero.

En el instante `Apartado A`, las tablas del sistema se encontrarán en este estado:

BCP (P)	
UID: 100	
GID: 100	
fd:	
0	teclado
1	monitor
2	monitor
3	2
4	1
5	0

BCP (H)	
UID: 100	
GID: 100	
fd:	
0	teclado
1	monitor
2	monitor
3	0
4	1
5	0

	i-nodo	**Posición**	**nrefs**	**Modo R/W**
1	678	11	2	10
2	748	24	1	01
3				
4				
5				

i-nodo	**nopens**	**Agrup.**	**Tamaño**
678	1	134	13
748	1	1012	8

b)

Partimos del estado de las tablas del apartado anterior. Tras el retorno de `fork`, el proceso H comienza a ejecutar la rama `case 0` del bloque condicional. La primera acción realizada es llamar al servicio `open` para crear o abrir el fichero "f" en modo sólo escritura. Puesto que ya ha sido creado por el proceso P previamente, lo abre conservando su contenido, puesto que no se emplea el flag `O_TRUNC`. Así pues, dado que el i-nodo ya se encuentra en la tabla de copias de i-nodos, el valor nopens de la entrada se ve incrementado en 1. Igualmente, se crea una entrada en la tabla intermedia (la número 3, la primera disponible) que apunta al i-nodo 748, inicializando el puntero a 0, el valor nrefs a 1 y el modo de apertura en sólo escritura. Por su parte, el descriptor utilizado por el proceso H para este fichero sería el primero disponible, es decir, el 3.

A continuación, el proceso H incrementa en 4 el valor del puntero correspondiente al fichero recién abierto mediante `lseek`. Puesto que se inicializó en 0, su valor pasa a ser 4. Inmediatamente después, el proceso H solicita leer 5 bytes del descriptor 4, pero sólo es posible leer 2 bytes, ya que el puntero se encuentra en la posición 11 y el fichero tiene un tamaño de 13 bytes. Así, el buffer `buf` contendrá los datos `BCcdefghijklmnopq`… y la variable `n` el valor 2. Finalmente, el proceso H escribe en el fichero "f" los dos bytes recién leídos a través del descriptor `fd2`. Es importante tener en cuenta que la posición del puntero para este descriptor es 4,

por lo que el contenido del fichero pasa a ser `7bcdBC9A`, es decir, se modifican los bytes quinto y sexto. El tamaño del fichero “f” no se ve alterado por esta escritura.

En el instante `Apartado B`, las tablas del sistema se encontrarán en este estado:

BCP (P)	**BCP (H)**
UID: 100	**UID:** 100
GID: 100	**GID:** 100
fd:	**fd:**

0	teclado
1	monitor
2	monitor
3	2
4	1
5	0

0	teclado
1	monitor
2	monitor
3	3
4	1
5	0

	i-nodo	**Posición**	**nrefs**	**Modo R/W**
1	678	13	2	10
2	748	24	1	01
3	748	6	1	01
4				
5				

i-nodo	**nopens**	**Agrup.**	**Tamaño**
678	1	134	13
748	2	1012	8

c)

En el instante `Apartado B`, el fichero “f” ocupara 8 bytes y tendrá el siguiente contenido, por los motivos expuestos en los dos anteriores apartados:

`7bcdBC9A`

d)

Esta es una posible alternativa, siendo posible alterar el orden de algunas de las llamadas:

```
        lseek(4, 0, SEEK_SET);
        dup2(4, 0);
        close(4);
        dup2(fd1, 1);
        close(fd1);
        execl("prog", "prog", NULL);
        exit(1);
```

Problema 2

Sea un Sistema de Ficheros de tipo UNIX con las siguientes características:

- Los i-nodos son de 128 bytes, con 10 punteros directos, 1 punteros indirecto simple y 1 puntero indirecto doble.
- Las agrupaciones son de 1KiB.
- Las direcciones son de 32 bits.
- Los mapas de bits de i-nodos y de agrupaciones ocupan 2 agrupaciones cada uno.

a) **[0.5 puntos]** Calcular el número máximo de ficheros y directorios que puede tener este sistema de ficheros.

Como el mapa de bits de agrupaciones ocupa 2 agrupaciones y cada agrupación es 1KiB, cada mapa tiene 16Ki bits. Por lo tanto, el número máximo de ficheros y directorios que pueden ser gestionados son 2^14.

b) **[0.5 puntos]** ¿Cuál sería el tamaño máximo en bytes del dispositivo que podría ser gestionado completamente con un sistema de ficheros como este? Seleccione una:

(a) 8 MiB
(b) 2 MiB
(c) 1 MiB
(d) 16 MiB
(e) 2 GiB
(f) 8 GiB

Los 2^14 bits del mapa de bits de agrupaciones permiten gestionar un dispositivo con ese número de agrupaciones. Como las agrupaciones son de 1Kib, el tamaño máximo serán 2^24 B. Opción (d)

c) **[1 punto]** Considerando solo la estructura del i-nodo, calcular en tamaño máximo en bytes de un fichero en este sistema de ficheros. Seleccione una:

(a) Aproximadamente 256 MiB
(b) Aproximadamente 256 GiB
(c) Aproximadamente 1 GiB
(d) Aproximadamente 16 GiB
(e) Aproximadamente 64 MiB
(f) Aproximadamente 128 MiB

Con agrupaciones de 1KiB y direcciones de 4B caben 2^8 direcciones por agrupación de indirección. Con el indirecto doble podremos direccionar archivos de más de (2^8)^2 agrupaciones, o 2^26 bytes. Opción (e)

d) **[1 punto]** ¿Qué sucede si se aumenta el tamaño de la agrupación de datos a 4KiB? Seleccione todas las respuestas verdaderas. Seleccione una o más de una:

(a) Se aumenta el tamaño máximo del fichero
(b) Se necesitan más accesos a disco para obtener los datos
(c) Aumenta la posibilidad de desperdicio de espacio (fragmentación interna)
(d) Disminuye el tamaño de dispositivo donde usar el sistema de ficheros
(e) Reduce el tiempo necesario para la lectura secuencial de los archivos

Opciones (a), (c) y (e).