Problemas de Sistemas Operativos

Facultad de Informática, UCM

Módulo 2: Sistemas de Ficheros

Problemas Básicos

1.-Este es un ejercicio práctico, en el que se pide al alumno que pruebe unos programas en una distribución cualquiera de Linux (por ejemplo la máquina virtual utilizada para el laboratorio). Vamos a experimentar el efecto del flag `O_APPEND` en la apertura de un fichero. Empezaremos por crear un fichero de texto llamado `fichero.txt` con un contenido aleatorio. Lo crearemos exáctamente escribiendo 134144 bytes en el fichero usando el siguiente filtro de la shell:

```
$ base64 /dev/urandom | head -c 134144 > fichero.txt
$
```

A continuación crearemos un programa progA.c en el mismo directorio, lo compilaremos y lo ejecutaremos. El contenido del programa será:

```c
//progA.c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(void)
{
    int i;
    int fd = open("fichero.txt", O_RDONLY);

    if (fd == -1) {
        return -1;
    }

    i = lseek( fd, 0, SEEK_END );
    printf("%d\n", i);

    return 0;
}
```

Cuando se ejecuta progA, ¿qué muestra el programa por su salida estándar? Consulta la página de manual de `lseek` para comprobar el valor que devuelve dicha llamada al sistema.

A continuación crea otro programa progB.c en el mismo directorio con el siguiente contenido:

```c
//progB.c
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

int main(void)
{
    char c = 'a';
    int fd = open("fichero.txt", O_WRONLY | O_CREAT | O_APPEND);
    if (fd == -1) {
        return -1;
    }
```

```
    lseek(fd, 1024, SEEK_CUR);
    write(fd, &c, 1);

    return 0;
}
```

Si lo compilamos y ejecutamos, ¿qué cambios provoca en `fichero.txt`? Puedes usar las utilidades `ls` y `diff` para comprobar dichos cambios. ¿Tiene algún efecto la llamada a `lseek` del programa progB.c?

Modifica ahora el programa progB.c eliminando el flag de apertura `O_APPEND`. ¿Cuál es la diferencia con el comportamiento anterior? ¿Tiene ahora algún efecto la llamada a `lseek`?

El flag `O_APPEND` garantiza que la operación de escritura y el desplazamiento del marcador al final del fichero son atómicas de forma global en el sistema, es decir, que si dos procesos están escribiendo al mismo tiempo en un fichero que han abierto con `O_APPEND` siempre añadirán información al final del fichero, independientemente del orden relativo de las distintas llamadas a open y a write.

2.-Juan crea un fichero de nombre *JFichero*. María crea un enlace físico a *JFichero* y le da el nombre *MFichero*. Juan elimina *JFichero*. Luego Juan crea un nuevo fichero y también le llama *JFichero*. ¿Cuántos ficheros diferentes existen después de las acciones anteriores? ¿Sería diferente la respuesta si el enlace que ha creado María fuese un enlace simbólico de nombre *MFichero* que apuntara a *JFichero*?

3.-Considerar un sistema donde el espacio libre se especifica empleando una lista enlazada de bloques libres. Suponer que se pierde el puntero a la lista. ¿Puede el sistema reconstruirla?

4.-Calcular el número de accesos a disco necesarios (para el caso peor y para el caso mejor) para leer 20 bloques lógicos consecutivos (no necesariamente los 20 primeros) de un fichero en un sistema con:

a) Asignación contigua (p.ej., ISO-9660)
b) Asignación no contigua mediante índice enlazado (FAT)
c) Asignación no contigua indexada (p.ej., UNIX)

Nota: Asumir que no hay ningún dato relacionado con el sistema de ficheros en memoria RAM y que el fichero se encuentra en el directorio actual.

5.-Un dispositivo de memoria flash de 64 MB de capacidad y bloques de 1KB, contiene un sistema de ficheros FAT. ¿Cuántos bytes son necesarios para almacenar la tabla FAT?

a. 64 KB
b. 128 KB
c. 1 MB
d. 512 KB
e. Ninguna de las respuestas anteriores es correcta

¿Es posible realizar enlaces rígidos en un sistema de ficheros tipo FAT? ¿Por qué no se puede establecer enlaces rígidos a ficheros de volúmenes distintos en sistemas de ficheros tipo EXT2?

6.-En la siguiente figura se representa una tabla FAT. Al borde de sus entradas se ha escrito, como ayuda de referencia, el número correspondiente al bloque en cuestión. También se ha representado la entrada de cierto directorio. Como simplificación del ejemplo, suponemos que en cada entrada del directorio se almacena: Nombre de archivo/directorio, el tipo (F=archivo, D=directorio), la fecha de creación y el número del bloque inicial.

Considere que el tamaño de bloque en este sistema de ficheros es 512 bytes y que el sistema operativo siempre selecciona el primer bloque libre (número inferior) cuando se crea un fichero o

# Bloque	Índice	# Bloque	Índice
1		10	
2		11	
3	15	12	
4		13	
5		14	
6		15	<EOF>
7		16	
8		17	
9		18	

Nombre	Tipo	Fecha	Num. Bloque
DATA	F	8-2-05	3

se redimensiona un fichero existente. En este supuesto, actualice el contenido de las estructuras del sistema de ficheros tras realizar cada una de las siguientes operaciones:

a) Creación del fichero DATA1 con fecha 3-1-10 y tamaño 10 bytes.
b) Creación del fichero DATA2 con fecha 3-2-10 y tamaño 1200 bytes.
c) El archivo DATA aumenta de tamaño, necesitando 2 bloques más.
d) Creación del directorio D con fecha 3-3-10 y tamaño 1 bloque.
e) Creación del fichero CARDS con fecha 3-12-10 y tamaño 2 Kbytes.

7.-Un sistema de ficheros UNIX utiliza bloques de 1024 bytes y direcciones de disco de 16 bits. Los nodos-i (entradas en una tabla que contiene la información descriptiva de los ficheros) contienen 8 direcciones de disco para bloques de datos, una dirección de bloque índice indirecto simple y una dirección de bloque índice indirecto doble. ¿Cuál es el tamaño máximo de un fichero en este sistema? ¿Y el de la partición?

8.-. Un programa UNIX crea un fichero e inmediatamente se posiciona (con `lseek()`) en el byte 55 millones. Luego escribe un byte.

a) ¿Cuántos bloques de disco ocupa ahora el fichero (incluyendo bloques indirectos)? Asúmase que los nodos-i contienen 10 índices directos, 1 índice indirecto simple, 1 índice indirecto doble y 1 índice indirecto triple, los bloques tienen un tamaño de 2 Kbytes y el tamaño de los índices (direcciones de bloques de disco) es de 32 bits.
b) ¿Qué sucesión de índices lógicos nos lleva al byte posicionado por lseek?

9.-Considere un sistema de ficheros basado en nodos-i en el que se utiliza un tamaño de bloque de 2KiB, un tamaño de puntero a bloque de 32 bits, y donde un nodo-i está formado por cuatro índices directos, 1 índice indirecto simple y 1 índice indirecto doble.

Un usuario crea un directorio en ese sistema, y sobre ese nuevo directorio (usándolo como directorio de trabajo) ejecuta secuencialmente los siguientes programas (primero programA y luego programB), que ya se encuentran compilados en otro directorio del sistema:

```
// programA
#include ...
#define BUFFER_SIZE 512

char buf[BUFFER_SIZE]="yyyyyy...y";

int main( void )
{
    int fd;

    fd=open( "file.txt", O_WRONLY | O_CREAT
                         | O_TRUNC, 0644 );
       write(fd,"x",1);
    lseek(fd, 8*BUFFER_SIZE, SEEK_CUR);
    write(fd,buf,BUFFER_SIZE);
    return 0;
}

// programB
#include ...
#define BUFFER_SIZE 512
```

```c
int main( void )
{
    int fd;
    char c;

    symlink("file.txt","anotherfile");
    fd=open("anotherfile", O_RDWR);
    lseek(fd, 16*BUFFER_SIZE, SEEK_END);
    write(fd,"z",1);
    lseek(fd, 0, SEEK_SET );
    read(fd, &c, 1);
    printf("%c\n",c);
    return 0;
}
```

Responda razonadamente a las siguientes preguntas, asumiendo que todas las llamadas al sistema que invocan los programas tienen éxito (no se valorarán respuestas no razonadas):

a) ¿Cuántos ficheros distintos se crean en el directorio?

b) ¿Qué se imprime por pantalla tras ejecutar ambos programas?

c) ¿Cuántos bloques ocupa en disco cada uno de los ficheros creados?

d) Suponga que solamente se ejecutara el programA pero sobre un sistema de ficheros con asignación enlazada (p.ej. tipo FAT), empleando el mismo tamaño de bloque, y trabajando sobre un directorio vacío. ¿Cuántos bloques ocuparía en disco el fichero creado por ese programa?

10.-Un sistema de ficheros basado en nodos-i y mapa de bits contiene la siguiente información:

Mapa de bits: 1 0 0 1 0 1 1 1 0 0 0 0 1 0 0 1 0 0 0

inode #2

Tamaño	1
#Enlaces	N/A
Tipo F/D	D
Directo	3
Indirecto	Null

nodo-i #3

Tamaño	2
#Enlaces	1
Tipo F/D	F
Directo	6
Indirecto	7

nodo-i #4

Tamaño	1
#Enlaces	
Tipo F/D	F
Directo	12
Indirecto	Null

nodo-i #5

Tamaño	
#Enlaces	N/A
Tipo F/D	D
Directo	0
Indirecto	Null

nodo-i #9

Tamaño	1
#Enlaces	N/A
Tipo F/D	D
Directo	
Indirecto	Null

bloque #0

.	5
..	2
C	9
D	4

bloque #3

.	2
..	
A	3
B	5
E	4

bloque #5

.	9
..	5

bloque #6: Datos sin formato

bloque #7

bloque #12

bloque #15: Datos sin formato

a) Rellene los huecos para que el sistema sea consistente. Asuma para ello que el tamaño se expresa en bloques.

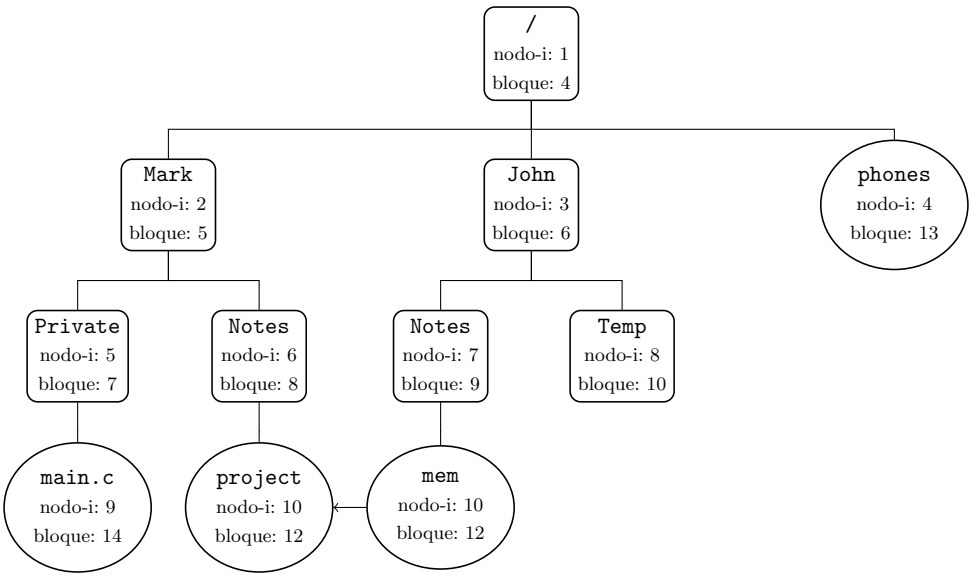
b) Dibuje el árbol del directorio empleando óvalos para los directorios, rectángulos para los ficheros y triángulos para los datos

11.-El sistema de ficheros de un SO diseñado a partir de UNIX utiliza bloques de disco de 1024 bytes de capacidad. Para el direccionamiento de estos bloques se utilizan punteros de 16 bits. Para indicar el tamaño del fichero y el desplazamiento (offset) de la posición en bytes en las operaciones read y write, se utilizan números de 64 bits. Cada nodo-i tiene 8 punteros de direccionamiento directo, 1 puntero indirecto simple y 1 puntero indirecto doble

a) ¿Cuál será el tamaño máximo de un fichero suponiendo despreciable el espacio ocupado por el superbloque y la tabla de nodos-i?

b) Si se modifica el tamaño de puntero pasándolo a 32 bits, ¿cuál será el nuevo tamaño máximo?

c) Dada la siguiente estructura de directorio[1] indicar los contenidos de los directorios y nodos-i que se encontrará el sistema (y el orden en que se los encontrará) al hacer la búsqueda del fichero *mem* desde el directorio raíz. Notese que *mem* es un enlace físico al fichero *project*.



12.-Un usuario desea dar formato a una partición de disco para almacenar su colección de fotografías. Cada fotografía se almacena en un fichero, con un tamaño (constante) de 7000 bytes.

El sistema de ficheros elegido está basado en i-nodos; la estructura de cada inodo está formada por 2 enlaces directos y 1 indirecto simple. Además, se usan 4 bytes para identificar un inodo, siendo el tamaño de puntero a bloque de 32 bits.

En el momento de proceder con el formateado del disco, se le ofrecen tres opciones distintas como tamaño de bloque a utilizar: 1024 bytes, 2048 bytes y 4096 bytes.

Ayude al usuario a tomar una decisión, contestando razonadamente a las siguientes preguntas:

a) ¿Cuál de las tres opciones presenta menos fragmentación interna? Para cada tamaño de bloque, indique el porcentaje de ocupación real de cada uno de los bloques de datos asignados a un fichero.

b) ¿Cuál de las tres opciones ofrece un mejor aprovechamiento de los bloques de disco disponibles? Para cada tamaño de bloque, indique qué porcentaje de bloques de disco se utilizarán para almacenar los datos de un determinado fichero.

c) ¿Cuál de las tres opciones ofrecerá, teóricamente, un mejor rendimiento de entrada/salida para la transferencia de un bloque de datos individual?

d) ¿Cuántos accesos a disco requerirá una lectura secuencial completa de una fotografía en cada caso, suponiendo que todas ellas se almacenan en un único directorio con nombre `fotografias` cuyas entradas se almacenan en un solo bloque de datos, y que la partición se montará en el directorio raíz (`/`) del sistema?

e) ¿Cómo afectará la selección de un tamaño de bloque determinado al tamaño de la tabla de inodos y al mapa de bits de bloques libres?

[1]Los directorios se muestran como rectángulos con bordes redondeados, y los ficheros como óvalos.

Problemas Adicionales

13.-Considerar un fichero que consta de 100 bloques. ¿Cuántas operaciones de disco son necesarias para cada una de las tres estrategias de asignación (contigua, enlazada e indexada) al realizar las siguientes operaciones?:

a) Añadir/Eliminar un bloque de información al comienzo
b) Añadirlo/Eliminarlo a la mitad
c) Añadirlo/Eliminarlo al final

14.-Dar 5 nombres de ruta diferentes para el fichero /etc/passwd. (Sugerencia: considerar las entradas de directorio "." y "..")

15.-¿Cuántos accesos a disco son necesarios para abrir el fichero games/chess en UNIX?

16.-Sugerir una razón por la cual alguien pudiera desear construir un directorio sobre los cuales los demás usuarios tuvieran permiso de ejecución pero no de lectura, en un sistema de ficheros de tipo UNIX.

17.-Dos estudiantes de informática, *Estudiante1* y *Estudiante2*, sostienen una discusión respecto a los nodos-i. *Estudiante1* argumenta que, dado que las memorias son cada vez más grandes y baratas, cuando se abre un fichero es más sencillo y rápido obtener una nueva copia del nodo-i para llevarla a la tabla de nodos-i en memoria, que buscar en la tabla entera para comprobar si ya está allí. *Estudiante2* discrepa. ¿Quién tiene razón?

18.-Un sistema de ficheros UNIX utiliza bloques de 512 bytes y direcciones de disco de 16 bits. Los nodos-i contienen 10 direcciones de disco para bloques de datos, una dirección de bloque índice indirecto simple y una dirección de bloque índice indirecto doble. Conteste de manera razonada a las siguientes cuestiones:

a) ¿Cuál es el tamaño máximo de un fichero en este sistema?
b) Un programa UNIX crea un fichero en este sistema e inmediatamente después escribe un byte de datos en la posición 1.000 y otro en la posición 10.000. ¿Cuántos bloques de datos ocupa este nuevo fichero en disco?

19.-Dado un sistema de ficheros tipo UNIX, en el que los directorios son relativamente pequeños (caben en un bloque de disco) y únicamente tiene una partición; razona qué operaciones de disco se necesitan para abrir el archivo "/usr/curso3/SO/alumnos.txt". Suponer que el nodo-i del directorio raíz está ya en memoria y que no se ha cargado en memoria anteriormente ningún otro elemento de la ruta.