Clarasdfgh
www.wuolah.com/student/Clarasdfgh

6002

RELACION-DEL-TEMA-4.pdf

RELACIÓN DEL TEMA 4

2º Sistemas Operativos

Grado en Ingeniería Informática

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación
Universidad de Granada

WUOLAH

El más PRO del lugar puedes ser Tú.

¿Quieres eliminar toda la publi de tus apuntes?

¡Fuera Publi!
Concéntrate al máximo

Apuntes a full.
Sin publi y sin gastar coins

Para los amantes de la inmediatez, para los que no desperdician ni un solo segundo de su tiempo o para los que dejan todo para el último día.

Quiero ser PRO

4,95 / mes

RELACIÓN DE PROBLEMAS TEMA 4

1.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.

> Los que tienen // delante venían dados
>
> Los que tienen # delante son editados

```
        FAT
1 | * (datos1)  10 | * (d)
2 | 4           11 |
3 | //15        12 |
4 | 5           13 |
5 | * (datos2)  14 |
6 | 7           15 | //* (datos) #16
7 | 8           16 | #17
8 | 9           17 | #* (datos)
9 | * (cartas)  18 |
```

Nombre	Tipo	Fecha	Nº bloque
Datos	F	2/2/90	3
Datos1	F	1/3/90	1
Datos2	F	2/3/90	4
D	D	3/3/90	10
Cartas	F	13/3/90	6

Tenga en cuenta que:

- el tamaño de bloque es de 512 bytes

- el asterisco indica último bloque

- todo lo que está en blanco en la figura está libre.

Rellene la figura para representar lo siguiente:

a) Creación del archivo DATOS1 con fecha 1-3-90, y tamaño de 10 bytes.

DATOS1 = 10B -> 1 bloque

b) Creación del archivo DATOS2 con fecha 2-3-90, y tamaño 1200 bytes.

DATOS2 = 1200B -> 3 bloques

c) El archivo DATOS aumenta de tamaño, necesitando 2 bloques más.

> Cambios indicados con #

d) Creación del directorio D, con fecha 3-3-90, y tamaño 1 bloque.

e) Creación del archivo CARTAS con fecha 13-3-90 y tamaño 2 kBytes.

CARTAS = 3KB -> 4 bloques

2.Si se pierde el primer puntero de la lista de espacio libre, ¿podría el Sistema Operativo reconstruirla? ¿Cómo?

Si se trata de una lista doblemente enlazada podemos recuperarla. Bastaría con recorrer la lista en orden inverso hasta llegar al puntero perdido.

También nos valdría tener una tabla de índices con el inicio de cada bloque.

3.El espacio libre en disco puede ser implementado usando una lista encadenada con agrupación o un mapa de bits. La dirección en disco requiere D bits. Sea un disco con B bloques, en que F están libres. ¿En qué condición la lista usa menos espacio que el mapa de bits?

El tamaño de una lista enlazada se puede determinar en función de los bloques, aunque no estén llenos. Los mapas de bits requieren que el bloque esté lleno. Cuando el tamaño de bloque es grande y no es fácil de llenar, la lista enlazada es más eficiente.

4.Entre los posibles atributos de un archivo, existe un bit que marca un archivo como temporal y por lo tanto esta sujeto a destrucción automática cuando el proceso acaba ¿Cuál es la razón de esto? Después de todo un proceso siempre puede destruir sus archivos, si así lo decide.

Porque este tipo de archivos suelen usarse para la ejecución de un programa. Al cerrarse este, es más conveniente que el archivo se "autodestruya" por su condición de temporal; que tener que hacer una llamada extra al terminar el programa para borrarlo.

5.Algunos SO proporcionan una llamada al sistema (RENAME) para dar un nombre nuevo a un archivo existente ¿Existe alguna diferencia entre utilizar esta llamada para renombrar un archivo y copiar el archivo a uno nuevo, con el nuevo nombre y destruyendo el antiguo?

La llamada RENAME no cambia el inodo, la copia sí.

6.Un i-nodo del sistemas de archivos s5fs de UNIX tiene 10 direcciones de disco para los diez primeros bloques de datos, y tres direcciones más para realizar una indexación a uno, dos y tres niveles. Si cada bloque índice tiene 256 direcciones de bloques de disco ¿cuál es el tamaño del mayor archivo que puede ser manejado, suponiendo que 1 bloque de disco es de 1KByte?

$10*1KB + 256*1KB + 256^2*1KB + 256^3*1KB =$

$5*2^{11} + 2^{18} + 2^{26} + 2^{34} =$

$16 * 2^{30} =$ **16GB**

7.Sobre conversión de direcciones lógicas dentro de un archivo a direcciones físicas de disco. Estamos utilizando la estrategia de indexación a tres niveles para asignar espacio en disco. Tenemos que el tamaño de bloque es igual a 512 bytes, y el tamaño de puntero es de 4 bytes. Se recibe la solicitud por parte de un proceso de usuario de leer el carácter número N de determinado archivo. Suponemos que ya hemos leído la entrada del directorio asociada a este archivo, es decir, tenemos en memoria los datos PRIMER-BLOQUE y TAMAÑO. Calcule la sucesión de direcciones de bloque que se leen hasta llegar al bloque de datos que posee el citado carácter.

Tamaño de bloque = 512 B

Tamaño de puntero = 4B

512/4 = 128 punteros/B

Tres accesos al disco

Los índices de cada n-ésimo bit buscado serán:

> i = n/128x128x512 -> n′ = n%128x128x512
>
> j = n′/128x512 -> n″ = n′%128x512
>
> k = n″/512 -> n‴ = n″%512

8.¿Qué organización de archivos elegiría para maximizar la eficiencia en términos de velocidad de acceso, uso del espacio de almacenamiento y facilidad de modificación (añadir/borrar /modificar), cuando los datos son:

a) modificados infrecuentemente, y accedidos frecuentemente de forma aleatoria

Contiguo, ya que no es necesario un método para modificar

b) modificados con frecuencia, y accedidos en su totalidad con cierta frecuencia

No contiguo, enlazado con tabla FAT: necesitamos un sistema que nos dé rápido acceso

c) modificados frecuentemente y accedidos aleatoriamente y frecuentemente

No contiguo, indexado: tiene la mejor velocidad de acceso aleatorio

9.Un controlador en software de disco duro recibe peticiones para los cilindros 10, 22, 20, 2, 40, 6 y 38 en ese orden. El brazo tarda 6ms en moverse de un cilindro al siguiente Suponiendo que el brazo está inicialmente en el cilindro 20 ¿cuánto tiempo de posicionamiento se requiere siguiendo los algoritmos

1.FCFS ?

20	10	22	20	2	40	6	38	TOTAL
/	60	72	12	108	228	204	192	**876ms**

2.SSTF ?

20	20	22	10	6	2	38	40	TOTAL
/	0	12	48	24	24	216	12	**336ms**

3.SCAN ?

20	22	38	40	20	10	6	2	TOTAL
/	12	96	12	126	60	24	24	**354ms**

10.¿Tendría sentido combinar en un mismo sistema de archivos dos métodos de asignación de espacio en disco diferentes? Por ejemplo, el método contiguo combinado con el enlazado. Ponga un ejemplo y razónelo.

Es práctico si cada uno ofrece ventajas a ficheros de diferentes características. Por ejemplo, el contiguo para la lectura secuencial de ficheros y el enlazado para la búsqueda de los ficheros e información específica.

11.¿Por qué la FAT en MS-DOS y Windows tiene un tamaño fijo en disco para cada sistema de archivos?

El tamaño máximo de la tabla FAT es de 4GB para cada aplicación: 2GB para esta, y 2GB para operaciones del kernel. Son 4GB porque en un sistema 32b tenemos 2^{32} localizaciones para cada byte de memoria. 2^{32} = 2.4 mil millones, lo que quiere decir que una dirección de memoria que tiene 32b de longitud, puede referirse a algo más de 4mil millones de direcciones, de ahí 4GB.

12.Para comprobar la consistencia de un sistema de archivos de Unix, el comprobador de consistencia (fsck) construye dos listas de contadores (cada contador mantiene información de un bloque de disco), la primera lista registra si el bloque está asignado a algún archivo y la segunda, si está libre. Según se muestra en la siguiente figura:

```
en uso:     1  0  1  0  0  1  0  1  1  0  1  0  0  1  0

libres:     0  0  0  1  1  1  0  0  0  1  0  1  1  0  1
```

¿Existen errores? ¿Son serios estos errores? ¿por qué? ¿qué acciones correctivas sería necesario realizar sobre la información del sistema de archivos?

Los bloques 2 y 7 no están ni en uso ni libres, son bloques pérdidos. No es un error grave, y lo soluciona el verificador añadiéndolos al final de la lista de libres

13.Cuando se abre el archivo /usr/ast/work/f, se necesitan varios accesos a disco. Calcule el número de accesos a disco requeridos (como máximo) bajo la suposición de que el i-nodo raíz ya se encuentra en memoria y que todos los directorios necesitan como máximo 1 bloque para almacenar los datos de sus archivos.

Directorio ROOT > inodo USR >> Directorio USR > inodo AST >> Directorio AST > Inodo WORK >> Directorio Work

6 accesos al disco hasta que encontramos el archivo /usr/ast/work/f

14.¿El planificador de Unix favorece a los procesos limitados por E/S (cortos) frente a los procesos limitados por CPU (largos)? Explique cómo lo hace. ¿Y el planificador de Windows 2000 o NT?

Sí. En la política de planificación de tiempo compartido en UNIX: el quantum de un proceso depende de su prioridad. La prioridad disminuye cuando se consume quantum y aumenta cuando un proceso se bloquea o tarda en usar la CPU. Cuando se bloquea (como en el caso de E/S), se le asigna al proceso una prioridad a nivel de núcleo.

En Windows 2000 o NT, los procesos E/S no tienen prioridad porque tienen su propio driver (módulo) y manager. Los procesos del kernel tienen mayor prioridad que los subsistemas del entorno.

15.Supongamos que un proceso, P1, abre el archivo "datos" en modo lectura/escritura y otro proceso, P2, abre el mismo archivo y con el mismo modo, y a continuación crea un proceso hijo que abre el archivo "/usr/pepe/doc" en modo lectura/escritura. Represente toda la información relevante sobre el estado de las tablas de descriptores de archivos, tabla de archivos y tabla de i-nodos después de dichas operaciones.

//TODO

16.Suponiendo una ejecución correcta de las siguientes órdenes en el sistema operativo UNIX:

```
/home/jgarcia/prog > ls -i  (* lista los archivos y sus números
  de i-nodos del directorio prog*)

18020                         fich1.c
18071                         fich2.c
18001                         pract1.c

/home/jgarcia/prog > cd ../tmp

/home/jgarcia/tmp > ln -s ../prog/pract1.c p1.c (* crea un
enlace simbólico *)

/home/jgarcia/tmp > ln ../prog/pract1.c p2.c     (* crea un
enlace absoluto o duro*)
```

represente gráficamente cómo y dónde quedaría reflejada y almacenada toda la información referente a la creación anterior de un enlace simbólico y absoluto ("hard") a un mismo archivo, pract1.c.

Dentro de directorio tmp creamos un enlace simbólico a p1.c y otro absoluto a p2.c:

- El enlace simbólico crea un nodo con un puntero a la estructura correspondiente del archivo con el que enlaza pract1.c.
- El enlace absoluto no crea inodo: se copia el puntero a la estructura con la que se relaciona pract1.c.

17.En Unix, ¿qué espacio total (en bytes) se requiere para almacenar la información sobre la localización física de un archivo que ocupa 3 Mbytes? Suponga que el tamaño de un bloque lógico es de 1 Kbytes y se utilizan direcciones de 4 bytes. Justifique la solución detalladamente.

Archivo de 3MB

Tam bloque -> 1KB

Tam direcciones -> 4B

2^{10} B/bloque / 2^2 B/dir = 256 dir/bloque

Archivo de 3MB = 3072B

3072 - 12 bloques directos = 3060

¿Cabe en el primer nivel? NO -> 3060-256 = 2804 y lo mandamos a direccionamiento

¿Cabe en el segundo nivel? SI, necesita 2804/256 = 11 bloques

Bloques totales: 11+1+1 = **13 bloques**

18. ¿Por qué en el sistema de archivos DOS (FAT) está limitado el número de archivos y/o directorios que descienden directamente del directorio raíz y en Unix no lo está?

Por la limitación de FAT del tamaño de los archivos: llega un punto en el cual una cadena de subdirectorios hace la ruta (y por tanto el nombre) del archivo mayor que el tamaño máximo (4GB, lo cual son muchísimos subdirectorios, pero hay que tener en cuenta que el nombre del archivo es sólo un atributo de este, también necesita almacenar datos).

19. ¿Cuál es el problema que se plantea cuando un proceso no realiza la llamada al sistema wait para cada uno de sus procesos hijos que han terminado su ejecución? ¿Qué efecto puede producir esto en el sistema?

El proceso no finaliza pero tampoco se vuelve a ejecutar: queda en estado zombie. Si además finaliza el padre pero no los hijos, estos quedan inalcanzables. Es una pérdida de recursos y podría llegar a causar problemas en sistemas de concurrencia.

20.Tenemos un archivo de 4 MBytes, calcule el espacio en disco necesario para almacenar el archivo (incluidos sus metadatos) en un Sistema de Archivos FAT32 y en otro s5fs. Suponga que el tamaño del cluster en FAT32 y el de bloque lógico en s5fs es de 4KBytes.

//TODO

21.En la mayoría de los sistemas operativos, el modelo para manejar un archivo es el siguiente:

-Abrir el archivo, que nos devuelve un descriptor de archivo asociado a él.

-Acceder al archivo a través de ese descriptor devuelto por el sistema.

¿Cuáles son las razones de hacerlo así? ¿Por qué no, por ejemplo, se especifica el archivo a manipular en cada operación que se realice sobre él?

Porque el nombre del archivo puede repetirse en varios directorios, y la ruta puede repetirse entre distintos usuarios (a parte de ser más engorroso de escribir). Con el descriptor nos referimos unívocamente al fichero deseado.

22.¿Cómo consigue optimizar FFS operaciones del tipo "ls –l" sobre un directorio? Explíquelo y ponga un ejemplo.

//TODO

23. Sea un directorio cualquiera en un sistema de archivos de Unix, por ejemplo, DirB. De él cuelgan algunos archivos que están en uso por uno o más procesos. ¿Es posible usar este directorio como punto de montaje? Justifíquelo.

Es posible utilizar el directorio como punto de montaje, pero los subdirectorios y archivos que se encuentren en él contarán como información del módulo montado y desaparecerán (se hacen invisibles) cuando desmontemos el dispositivo. Si están en uso mientras hacemos esta operación existe riesgo de error.

24.¿Es correcto el siguiente programa en UNIX, en el sentido de que no genera inconsistencias en las estructuras de datos que representan el estado de los archivos en el sistema? Represente con un esquema gráfico la información almacenada en las estructuras (tablas de descriptores de archivos, tabla de archivos y tabla de i-nodos) que representan el estado real referente a los archivos utilizados por los siguientes procesos en el instante de tiempo en que:

el PROCESO HIJO DE A ha ejecutado la instrucción marcada con (1),

el PADRE (PROCESO A) ha ejecutado la (2),

y el PROCESO B ha ejecutado la (3).

Para ello suponga que son los únicos procesos ejecutándose en el sistema en ese momento.

```
/* PROCESO A */
main(){
    int archivos[2], pid;
    archivos[0]=open("/users/prog.c",O_RDONLY);
    archivos[1]=open("/users/prog.c",O_RDWR);

    if ((pid=fork())==-1)
        perror("Error en la creación del proceso);

    else if (pid==0)
        execlp ("ls", "ls", NULL); /*(1)*/

    else
        close(archivos[0]); /*(2)*/
    }

/* PROCESO B */
main(){
    int archivos[2];
    archivos[0]=open("/users/prog.c",O_RDONLY);
    archivos[1]=open("/users/texto.txt",O_RDWR);
    dup(archivos[0]; /*(3)*/
}
```

//TODO