1

SISTEMA DE FICHEROS

Problema 1.1

La cola de acceso a un disco tiene las siguientes peticiones (solamente indicaremos el cilindro afectado). 10, 200, 75, 32, 450, 123.

El tiempo de búsqueda es de 3 + 0,04·x ms, siendo x el número de cilindros a recorrer.

Suponiendo que el disco se encuentra inicialmente en el cilindro 300 calcular la suma de los tiempos de búsqueda para las siguientes políticas de planificación: FCFS, SSF y CSCAN

Problema 1.2

Sea un sistema de ficheros UNIX de direcciones de bloque de 32 bits.

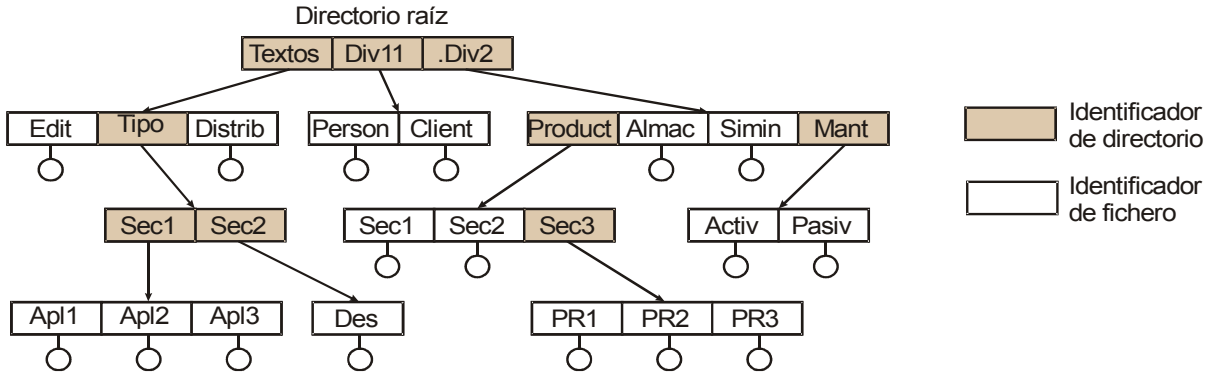
Determinar el tamaño del máximo fichero para los siguientes casos: a) Bloque: 512 B, b) Bloque 4 KB

Suponiendo que el nodo-i del fichero es el único dato existente en memoria y que estamos en el caso de bloques de 4KB, determinar el número de accesos a disco que son necesarios, en el peor caso, para acceder al byte situado en la posición 3 Mbytes.

Problema 1.3

Sobre el árbol de directorios de la figura, indicar el número de accesos al disco necesario para realizar la operación de apertura del fichero /Textos/Tipo/Sec1/Apl3 para su lectura en los siguientes supuestos:

- *No existe ninguna información en la cache de bloques.*
- *Se acaba de abrir el fichero /Textos/Tipo/Sec2/Des con permisos de lectura y escrituras.*



Problema 1.4

Se tiene un disco con sectores de 512 B y una capacidad de 150 GB. Se quiere dar formato a un sistema de ficheros tipo UNIX para dicho disco. Suponiendo que:

- *El tamaño del bloque es de 4 KB*
- *El tamaño medio de los ficheros a almacenar es de 100 KB.*

Determinar el tamaño del nodo_i así como los tamaños de las zonas que componen el sistema de ficheros.

Determinar con precisión el tamaño máximo que podría tener un fichero en sistema.

Problema 1.5

Realizar una figura con las estructuras de información relativas al servidor de ficheros que existen después de la secuencia siguiente:

- *Proceso 1 Abre fichero /Textos/Tipo/Ses1/Apl3 para lectura. Dicho fichero tiene inicialmente un tamaño de 45.567 B.*
- *Proceso 2 Abre fichero /Textos/Tipo/Ses1/Apl3 para lectura y escritura.*
- *Proceso 1 lee 400 B de Apl3.*
- *Proceso 2 Adelanta el puntero de posición de Apl3 en 20.*
- *Proceso 1 Crea un hijo, que será el proceso 3.*
- *Proceso 3 Escribe 12.560 B en Apl3.*

Problema 1.6

Un fichero /home/juan/dat.txt tiene inicialmente el siguiente contenido, expresado como bytes en hexadecimal:

01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13

Se realiza la siguiente secuencia

- Proceso 1 abre el fichero `fd = open("/home/juan/dat.txt", O_RDWR);`
- Proceso 2 abre el fichero `fd = open("/home/juan/dat.txt", O_RDWR);`
- Proceso 1 crea un hijo, el proceso 3
- Proceso 2 ejecuta `read(fd, buf, 5);`
- Proceso 3 ejecuta `lseek(fd,2,SEEK_CUR);`
- Proceso 3 ejecuta `write(fd, "FGTHIJ", 6);`
- Proceso 2 ejecuta `lseek(fd,-9, SEEK_END);`
- Proceso 2 ejecuta `write(fd, "Esto es una prueba para ver que contenido", 35);`

Indicar el contenido final del fichero, así como su tamaño.

Problema 1.7

El Sistema Operativo UNIX, gracias a la operación de montaje de una partición (que ha de contener un sistema de ficheros simple), ofrece la visión de un único árbol de ficheros que realmente está soportado sobre varias particiones. Este diseño plantea las siguientes limitaciones:

- *No se puede realizar un enlace físico (link) entre dos nombres de fichero cuando pertenecen a particiones distintas.*
- *Un fichero puede crecer como máximo hasta agotar el espacio de la partición a la que pertenece, pero no puede extenderse sobre otras particiones, a pesar de que quede espacio libre en estas.*

Se pide que, para cada una de estas limitaciones, discuta el porqué de la misma indicando las estructuras de datos de los sistemas de ficheros que están involucradas.

Problema 1.8

Un determinado computador personal dispone de un único disco duro con dos Sistemas Operativos, un Windows y un Linux. El gestor de arranque consulta al usuario con cuál Sistema Operativo desea arrancar en cada momento. El disco duro está particionado como se muestra en la siguiente tabla, donde el campo punto de montaje indica en que directorio se procede a montar esa partición cuando se arranca en Linux.

Partición	***Tamaño***	***Tipo de SF***	***Punto de montaje***
`/dev/sda1`	*100 GiB*	*FAT32*	`/windows`
`/dev/sda2`	*48 GiB*	*Linux UFS*	`/`
`/dev/sda3`	*8 GiB*	*Linux swap*	
`/dev/sda4`	*100 GiB*	*Linus UFS*	`/home`

Considere que hemos arrancado en Linux y ejecutamos el mandato:

```
cp /tmp/UnGiga.dat /home/C:/usuario/
```

Siendo "C:" un enlace simbólico a "`/windows/Documents and Settings/`" (espacio para datos de las cuentas de usuario en Windows), existiendo todas las rutas necesarias, y disponiendo el usuario de todos los derechos necesarios.

a) *Determine las secuencias de directorios que será necesario acceder durante la fase de decodificación de las rutas de origen y destino de este mandato (llamadas open), indicando a cada paso el criterio para decidir pasar de cada uno al siguiente.*

b) *Identifique qué datos y qué metadatos deberán ser manipulados (leídos o escritos) durante el bucle de copia de datos propiamente dicho.*

Problema 1.9

Un sistema de ficheros similar al de UNIX, presenta las siguientes características:

Representación de ficheros mediante nodos-i con 12 direcciones directas a bloque, un indirecto simple, un indirecto doble y un indirecto triple y direcciones de bloques de 4 bytes. El tamaño del bloque del sistema de ficheros es de 8 KB.

El sistema de ficheros emplea un cache de 4 MB con una política de reemplazo LRU.

*Sobre este **sistema** se ejecuta el siguiente fragmento de código:*

```
#define DATA_SIZE     2048
#define N             5000
char buffer[DATA_SIZE];
int fd, i;

fd = open("fichero", O_RDWR);
for (i = 0 ; i < N; i++)
   write(fd, buffer, DATA_SIZE);
lseek(fd, 0, SEEK_SET);
for (i = 0 ; i < N; i++)
   read(fd, buffer, DATA_SIZE);
Close(fd);
```

Teniendo en cuenta que el fichero tiene un tamaño inicial de 16 MB, que la cache se encuentra inicialmente vacía y que no se realiza ninguna operación de volcado (flush) durante el mencionado programa, se pide:

a) ¿Cuántos accesos a disco se producen durante los bucles de lectura y escritura utilizando una política de actualización write-through (escritura inmediata)?

b) ¿Cuántos accesos se producirían si se utilizase una política de actualización write-back (escritura diferida)?

c) Calcule la tasa de aciertos a la cache que se produce durante el bucle de lectura.

Problema 1.10

Sean los códigos programa 1 y programa 2 que ejecutará el usuario jperez

```
/* Programa 1 */
int main (void) {
int fd; int pid; int cont;

fd = open ("/alfa", O_RDWR);
cont = write(fd, "Escribo alfa", strlen("Escribo alfa"));
close(fd);
fd = open ("/beta/b4/tyz", O_RDWR);
lseek(fd, 13,SEEK_SET);
cont = write(fd, "del ejemplo", strlen("del ejemplo"));
pid = fork();
/* PUNTO A */
switch (pid) {
   case -1:
      break;
   case 0: /* hijo */
      cont = write(fd, "del hijo", strlen("del hijo"));
      /* PUNTO C */
      break;
   default: /* padre */
      cont = write(fd, "del padre", strlen("del padre"));
   /* PUNTO D */
   }
return 0;
}

/* Programa 2 */
int main (void) {
int fd; int cont;
fd = open ("/beta/b4/tyz", O_RDWR);
cont = write(fd, "Proceso 2 escribe", strlen("Proceso 2 escribe"));
/* PUNTO B */
```

```
close(fd);
return 0;
}
```

Suponiendo que el orden de ejecución es el siguiente:

- *Proceso padre con código programa 1 ejecuta hasta el punto A*
- *El proceso con código programa 2 ejecuta hasta el punto B*
- *El proceso hijo con código programa 1 ejecuta hasta el punto C*
- *Vuelve a ejecutar el padre hasta el punto D*

Y que los contenidos de algunas agrupaciones y nodos-i son los mostrados en las tablas adjuntas, se pide:

a) *Establecer el esquema de directorios.*
b) *Indicar el tamaño de la agrupación*
c) *Indicar, para cada uno de los procesos, los valores devueltos por cada uno de los servicios del SO que ejecuta.*
d) *Indicar el contenido del fichero /alfa en el punto A y el contenido del fichero /beta/b4/tyz en los puntos A, B y C.*

Agrupación	***Contenido***
32	*. 2; .. 2; usr 121; alfa 243; bi 453; beta 54*
73	*. 417; .. 453 .....*
74	*. 628; .. 54; abc 742; tyz 375*
112	*. 412; .. 453 .....*
138	*. 453; .. 2; bi 430; apli1 412; apli2 417*
255	*¿Cómo se aplica la sección 3.3 del plan de seguridad .....*
271	*. 121; .. 2; prof1 301; prof2 317; prof3 371*
1300	*. 301; .. 121 .....*
2342	*CALLE SANDOVAL, 3 ......*
3207	*. 430; .. 453 .....*
3765	*Desde el punto de vista del usuario, la tabla Panel ....*
4731	*El usuario jramirez está .....*
6734	*. 54; .. 2: b1 621; b2 612; b3 614; b4 628*
6777	*En el esquema correspondiente se representan estas líneas .....*
15301	*Para el ejemplo de prueba se han propuesto 5 casos que ....*

Nodo-i	***Dueño (UID)***	***Permisos***	***Tipo***	***Agrupaciones***	***Tamaño***	***SUID, SGID***
2	*Root (0)*	*rwx r-x r-x*	*directorio*	*32*	*180*	
54	*Root (0)*	*rwx r-x r-x*	*directorio*	*6734*	*180*	
121	*Root (0)*	*rwx r-x r-x*	*directorio*	*271*	*150*	
243	*Root (0)*	*rwx —- —-*	*usuario*	*4731*	*653*	
301	*jperez*	*rwx r-x r-x*	*directorio*	*1300*	*300*	
317	*lgomez*	*rwx r-x r-x*	*usuario*	*255 y 257*	*1568*	
371	*rfung*	*rwx r-x r-x*	*usuario*	*15301 a 15305*	*4354*	
375	*jperez*	*rwx r-x r-x*	*usuario*	*6777 a 6779*	*2564*	
412	*Root (0)*	*rwx r-x r-x*	*directorio*	*112*	*120*	
417	*Root (0)*	*rwx r-x r-x*	*directorio*	*73*	*180*	
430	*Root (0)*	*rwx r-x r-x*	*directorio*	*3207*	*330*	
453	*Root (0)*	*rwx r-x r-x*	*directorio*	*138*	*150*	
612	*lgomez*	*rwx r-x r-x*	*usuario*	*3765*	*473*	
614	*lgomez*	*rwx r-x r-x*	*usuario*	*4321 y 486*	*1324*	
621	*lgomez*	*rwx r-x r-x*	*usuario*	*2342*	*875*	

628	*jperez*	*rwx r-x r-x*	*directorio*	*74*	*120*	
742	*jperez*	*rwx r-x r-x*	*usuario*	*5701 a 5708*	*7432*	

Problema 1.11

Sea un sistema que utiliza páginas de 4KB y cuyos bloques de sistema de ficheros son de 4KB. En este sistema existe un fichero denominado 'fichpru' de 8742 B que ocupa, por ese orden, los bloques 10004, 10007 y 10001.

a) *Recordando la forma en que el sistema de ficheros accede al disco, indicar las operaciones que hace el sistema de ficheros sobre el disco para atender el servicio:* `write(fd, "X", 1);`. *Supondremos que el sistema de ficheros no tiene cache, por lo que hace todas las operaciones de lectura y escritura directamente sobre disco.*

b) *Sean los dos programas adjuntos, que sobrescriben las 3000 primeras posiciones pares del fichero, respetando el valor original de las impares.*

Seguimos suponiendo que el sistema de ficheros no tiene cache.

Para cada uno de los programas de la tabla 1, indicar por orden los 6 primeros accesos a disco que se generan dentro del bucle `for` *(externo para el programa 2). Detallar el bloque accedido y el tipo de acceso.*

Programa 1

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
int main(void)
{
   int fd, i;
   fd = open("fichpru", O_WRONLY);
   for (i=0; i<3000; i++) {
      write(fd, "X", 1);
      lseek(fd, +1, SEEK_CUR);
   }
   close(fd);
   return 0;
}
```

Programa 2

```
#include <sys/stat.h>
#include <sys/types.h>
#include <fcntl.h>
#include <unistd.h>
int main(void)
{
   int fd, i, j;
   char buff[20];
   fd = open("fichpru", O_RDWR);
   for (i=0; i<300; i++) {
      read(fd, buff, 20);
      for (j=0; j<10; j++)
         buff[2*j] = 'X';
      lseek(fd, -20, SEEK_CUR);
      write(fd, buff, 20);
   }
   close(fd);
   return 0;
}
```

Tabla 1

Problema 1.12

Se desea implementar un spooler, un programa que recorre un directorio de trabajos pendientes (`/var/spool/works/`*) y los envía a otro programa que los ejecuta. Cuando se desea que un trabajo se lance por medio del spooler se escribe un nuevo fichero en el directorio compartido, que este proceso inspeccionará periódicamente. Las características del funcionamiento del programa spooler son las siguientes:*

- *Durante el recorrido del directorio se comprueba si hay nuevas entradas (saltándose las entradas '.' y '..'). Esta operación la realizará la función* `recorrer_directorio`.

- *La verificación de si hay trabajos pendientes en el directorio se ejecuta cada segundo. La función `main` programará la temporización y preparará a la función anterior (`recorrer_directorio`) para que se ejecute en ese momento.*
- *Por cada fichero encontrado en el directorio se debe ejecutar el programa `/bin/ejecutor`, al que se le pasará por su entrada estándar el contenido del fichero. La ejecución de este programa se hará desde la función `procesar_fichero`.*
- *El fichero de un trabajo, una vez procesado, se debe borrar. Esto también lo hará la función `procesar_fichero`.*

a) *Implementar la función `recorrer_directorio`*

b) *Implementar la función `procesar_fichero`.*

c) *En la escritura y posterior borrado de trabajos se plantea otro problema. Los usuarios que escriben trabajos en el directorio no tienen por qué ser los mismos que el usuario que ejecuta el spooler (que será, probablemente, un usuario del sistema llamado `spool`). Supongamos que los usuarios usan un programa llamado submit, para programar trabajos (escribirlos en el directorio), y que el procesamiento es por medio del programa spooler que ya hemos visto. ¿Cuáles serían las consideraciones sobre propietarios y permisos que habría que tener en cuenta en los ficheros, directorios y programas que participan?*

Problema 1.13

Sea el siguiente programa de nombre "`c0p1`" para copiar archivos (por simplicidad se ha eliminado todo control de errores):

```
 1      /* c0p1 origen destino
 2       * Copia "origen" sobre "destino" byte a byte,
 3       * haciendo uso de los descriptores 0 y 1.
 4       */
 5      int main(int argc char * argv[])
 6      {     unsigned char byte[1];
 7            close(0); open(argv[1], O_RDONLY);
 8            close(1); open(argv[2], O_WRONLY|O_CREAT|O_TRUNC, 0666);
 9            while(write(1,byte,read(0,byte,1))>0)
10                 continue;
11            return 0;
12      }
```

Se pide:

a) *Describa claramente la evolución de los descriptores 0 y 1 del proceso durante la ejecución de las líneas 7 y 8. Para ello, dibuje las tablas internas que el Sistema Operativo utiliza que para gestionar los descriptores de fichero (comenzando en el BCP del proceso **`c0p1`** y terminando en los nodos-i involucrados). Explique el objetivo de estas tablas y muestre el contenido de sus campos más relevantes.*

*A continuación, y razonando sobre las tablas y campos mostrados en el apartado anterior, explique claramente el comportamiento y el resultado final de la ejecución del mandato **`c0p1 origen destino`** en los supuestos siguientes:*

b) ***`origen`** y **`destino`** son dos ficheros convencionales distintos de tamaños 1MB y 2MB respectivamente.*

c) ***`origen`** y **`destino`** son dos enlaces simbólicos distintos que apuntan a un mismo nombre (**`fichero`**), que es un fichero convencional de tamaño 4MB.*

d) ***`origen`** y **`destino`** son dos enlaces simbólicos distintos que apuntan a un mismo nombre (**`terminal`**), que fue creado como enlace duro (hard link) al fichero especial orientado a carácter **`/dev/tty`** que representa el terminal de control asociado al proceso.*

*Por último, y volviendo a considerar las condiciones del apartado **B**, y que el Sistema de Ficheros está basado en nodos-i con direcciones de 32 bits y bloques de 2KB y que se utiliza una cache de bloques con capacidad para 1Mbyte, política de reemplazo LRU, delayed-write para datos y write-through para metadatos, conteste:*

e) Sin contar los accesos necesarios para la decodificación de los nombres de los ficheros,

1. *¿Cuántos accesos a disco ocurrirán durante la ejecución completa del mandato?*
2. *¿Cuál será el contenido detallado de la cache en el instante de terminación del proceso?*