enero2020.pdf

TuTorMentor

Sistemas Operativos

3º Grado en Ingeniería Informática

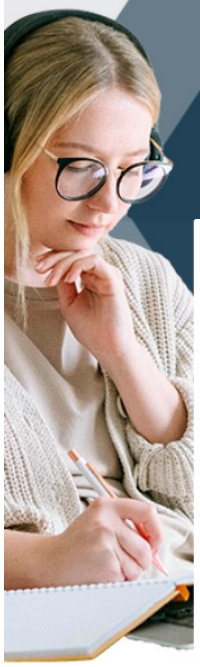
Facultad de Informática
Universidad Complutense de Madrid

Maths informática

CURSOS INTENSIVOS PARA EXÁMENES DE
CONVOCATORIA ORDINARIA
Y EXTRAORDINARIA

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

CURSOS INTENSIVOS PARA EXÁMENES DE
CONVOCATORIA ORDINARIA
Y EXTRAORDINARIA



Academia especializada en estudios
de la Facultad de Informática
Maths informática

SISTEMAS OPERATIVOS - CUESTIONES
13 de enero de 2020

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Cuestión 1. (1.5 puntos)

Considere el fragmento de código que implementa un módulo del kernel Linux que se muestra en la columna de la izquierda, y el programa de usuario que se muestra en la columna de la derecha:

Extracto del módulo del kernel

```
static int Device_Open = 0; // Variable global.
static int device_open(struct inode *ino, struct file *f) {
  if (Device_Open) return -EBUSY;
  Device_Open++;
  try_module_get(THIS_MODULE);
  return SUCCESS;
}

static int device_release(struct inode *ino, struct file *f)
{
  Device_Open--;
  module_put(THIS_MODULE);
  return 0;
}

static ssize_t device_read(struct file *filp, char *buffer,
size_t length, loff_t * offset) {
  char msg[] = "Hola, mundo";
  printf("Ejecutando operación de lectura"); //printk()
  buffer = msg;
} ... copy_to_user()
// devuelven la # leídos
static struct file_operations fops = {
  .read = device_read,
  .open = device_open,
  .release = device_release };

// Implementación de init_module y cleanup_module
// ...
```

Programa de usuario

```
#include <stdio.h>
/* Resto de includes */
...
int main( int argc, char * argv[] )
{
  int fd = 0; int bytes = 0;
  int size = atoi( argv[ 2 ]);

  char * buffer = (char *) malloc( size );

  if( buffer == NULL ) {
        printf( "Error en reserva\n" ); return -1;
  }

  fd = open( argv[1], O_RDONLY );

  if( fd == -1 ) {
        printf( "Error en apertura\n" ); return -1;
  }

  bytes = read( fd, buffer, size );

  if( bytes == -1 ) {
        printf( "Error leyendo\n" ); return -1;
  }

  printf( "%s\n", buffer );
  printf( "Lei %d bytes\n", bytes );

  free( buffer ); close( fd );
}
```

Suponga que:

1. Las funciones `init_module` y `cleanup_module` del módulo han sido correctamente implementadas.
2. Tras la correcta carga del módulo, la función `init_module` ha obtenido los números *major* y *minor* **237** y **0**, respectivamente, y ha asociado la estructura `fops` correctamente.
3. Se desea que, ante una lectura en un dispositivo asociado al módulo, se devuelvan al usuario los primeros `length` bytes de la cadena *"Hola, mundo"* (o el tamaño de la misma en caso de ser éste menor), y se muestre el mensaje *"Ejecutando operación de lectura"* por la salida de *log* del kernel.
4. La salida de la orden `ls -l /dev/examen*` en el sistema es la siguiente:

```
crw------- 1 root root 237, 0 ene  3 14:17 /dev/examen2
crw-rw-rw- 1 root root 237, 0 ene  3 14:17 /dev/examen
```

Responda razonadamente a las siguientes preguntas:

a) (0.5 pt) Explique cuál es el cometido de la invocación a las funciones `try_module_get` y `module_put`, y de la variable `Device_open`. ¿Cómo puede consultarse el valor del contador desde la línea de órdenes?

Previene el borrado de un módulo que esta en uso.
Incrementa/decrementa el contador interno del kernel para cada módulo.
Si el contador es distinto de 0 no puede eliminarse.

$ lsmod (en la columna Used by)

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

WUOLAH
Escaneado con CamScanner

b) (0.5 pt) Identifique los errores cometidos por el programador en la implementación de device_read y proponga una solución correcta que cumpla con las especificaciones del punto 3 anterior.

```
static ssize_t device_read (struct file * filp, char * buffer, size_t length,
                            loff_t * offset) {
    char msg[] = "Hola, mundo";
    int bytes_read = min (length, strlen (msg)) // calculamos el numero para copia
    if (copy_to_user (buffer, msg, bytes_read)) // delegamos usar copy_to_user()
        return -EFAULT;
    printk (KERN_INFO "the driver, create a dev file with %u", // usamos printk()
    return bytes_read; // devolvemos los bytes leidos
}
```

copy_to_user / copy_from_user } copia privada de los datos en el espacio del kernel. No podemos alterar en los punteros del espacio de usuario, el driver puede no tener acceso

c) (0.5 pt) Indique la salida tras la ejecución por parte del usuario usuarioso de cada una de las siguientes órdenes considerando la implementación correcta de la función device_read:

```
$ ./usuario /dev/examen 1      H
                               Leí 1 Bytes
$ ./usuario /dev/examen 4      Hola
                               Leí 4 Bytes
$ ./usuario /dev/examen 16     Hola, mundo
                               Leí 16 Bytes
$ ./usuario /dev/examen2 16
  "Error en apertura" [no tenemos permisos]
```

Cuestión 2. (1.5 puntos)

Considere un sistema monoprocesador con una política de planificación de procesos de 2 niveles con realimentación. El nivel L1 usa una política de planificación expropiativa *circular* (round robin) con quanto de tiempo 2. El nivel L2 usa una planificación expropiativa *primero el de menor tiempo restante* (SRTF). Al principio hay 3 procesos en la cola del nivel L1: P1, P2 y P3, en este orden. Los patrones de ejecución de los procesos son los siguientes (y se repiten indefinidamente):

P1 (5-CPU,5-E/S), P2 (4-CPU,5-E/S), P3 (6-CPU, 5-E/S)

La cola L2 está inicialmente vacía. Cuando un proceso agota su tiempo de CPU, pasa a la cola de nivel inferior. Cuando acaba una operación de E/S, los procesos entran en la cola de mayor prioridad. Usando el diagrama de tiempos de la figura, muestre:

a. Qué proceso está ejecutándose (utilice 'X1' o 'X2' para marcar el proceso en ejecución desde el nivel L1 ó L2, respectivamente, y 'O' para marcar el proceso bloqueado por E/S) y qué procesos hay en cada nivel (utilice las filas de L1 para el primer nivel y las de L2 para el segundo), durante las 20 primeras unidades de tiempo.
b. Calcule el porcentaje de utilización de CPU y los tiempos de espera de cada proceso.

	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20
P1(5,5)	X1	X1							X2	X2	X2	—	—	—	—	—	X1	X1			
P2(4,5)			X1	X1			X2	X2	—	—	—	—	—	X1	X1					X2	X2
P3(6,5)					X1	X1						X2	X2			X2			X2	—	—
L1 RR(2)	P2	P2	P3	P3																	
	P3																				
L2 SRTF			P1	P1	P2	P2	P1	P1	P3	P3	P3			P3	P3	P2	P3	P3	P2	P1	P1
					P1	P1	P3	P3									P2	P2	P1		

Porcentaje de uso de CPU: 100 %
Tiempo de espera de P1: 9 unidades
Tiempo de espera de P2: 8 unidades
Tiempo de espera de P3: 13 unidades

en el SRTF también puede acabar la tarea

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Escaneado con CamScanner

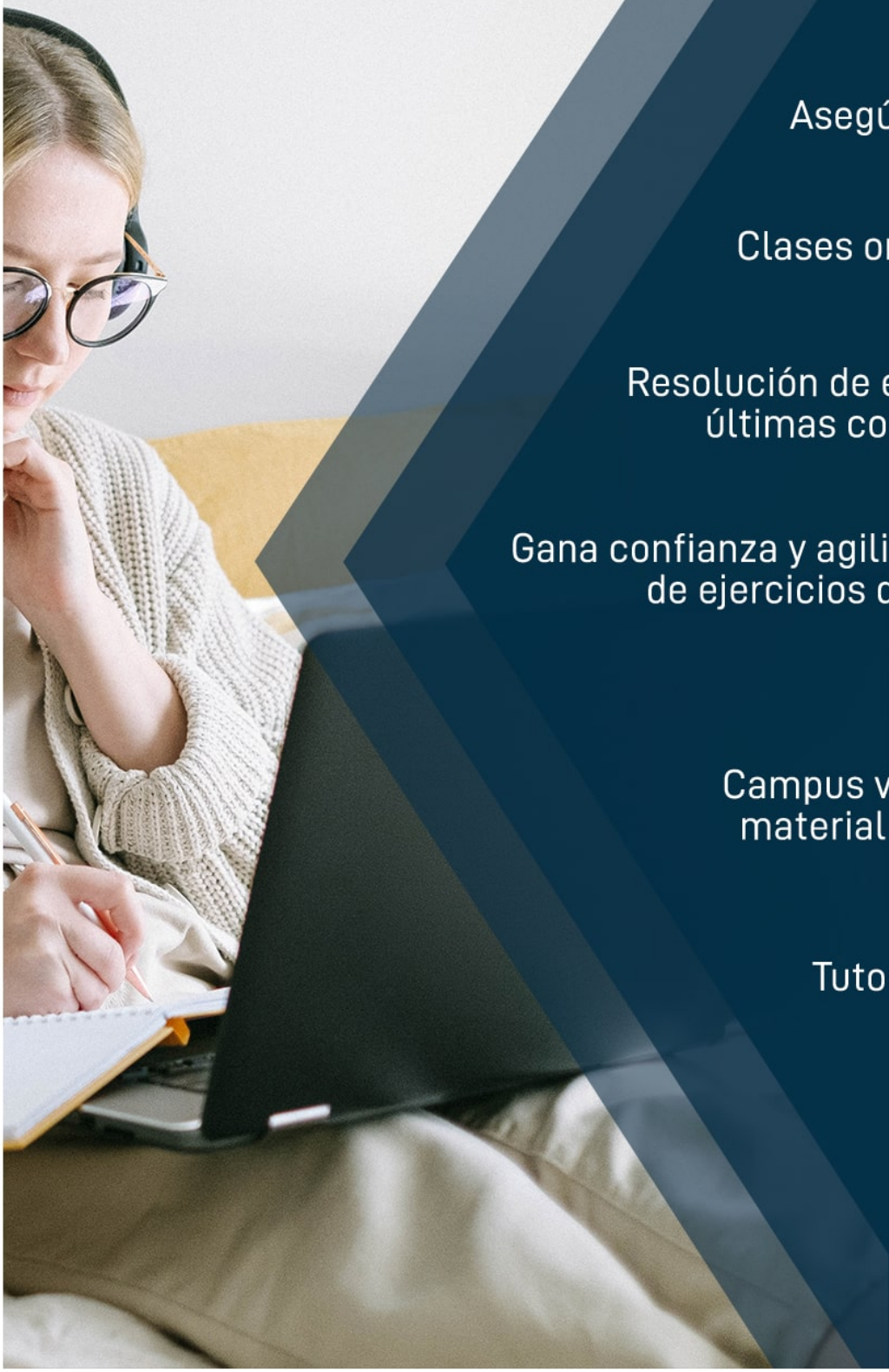
¡Adiós a las imperfecciones de la piel con My Clarins! Habla con el experto en CLARINS.COM

Maths informática

Academia especializada en estudios de la Facultad de Informática

CURSOS INTENSIVOS PARA EXÁMENES DE

CONVOCATORIA ORDINARIA Y EXTRAORDINARIA



Asegúrate un aprobado

Clases online en directo

Resolución de exámenes de las últimas convocatorias

Gana confianza y agilidad en la resolución de ejercicios de exámenes

Campus virtual con material de ayuda

Tutorías y resolución de dudas

www.mathsinformatica.com

Cuestión 3. (1.5 puntos)

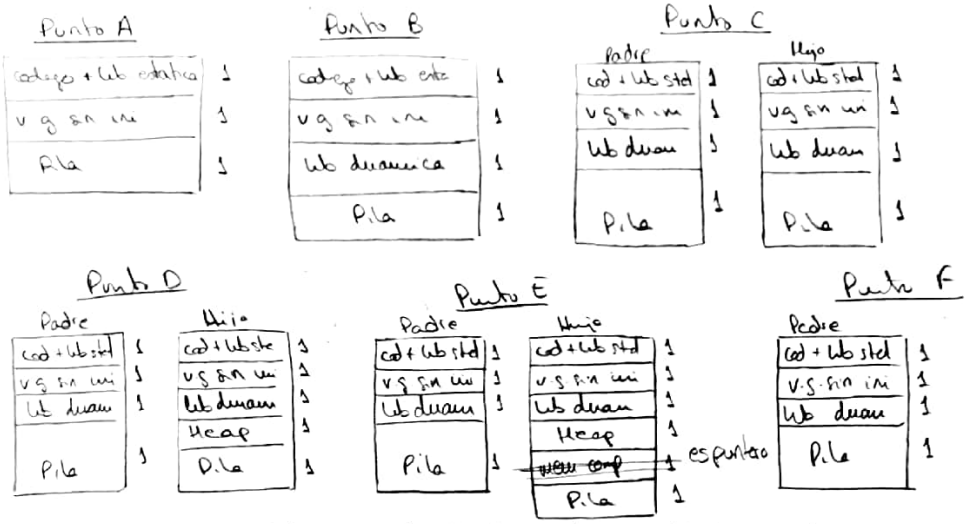
Un sistema utiliza gestión de memoria virtual con paginación bajo demanda con páginas de 4KB, direcciones de memoria de 32 bits y memoria principal de 8G. Se quiere ejecutar en dicho sistema el siguiente programa:

```
#include <stdio.h>
/* resto de includes */
...
#define M 512
int array[M];
int *copia;

int main(int argc, char *argv[]) {
  int fi,fo;
  pid_t pid=0;
// Punto A
  fd=open("file",O_RDONLY));
// Punto B
  pid=fork()
// Punto C
  if(pid==0){
    copia=(int *) malloc(M*sizeof(int));
// Punto D
   read(fd, array, M*sizeof(int));
   memcpy(copia,array,M);
// Punto E
    free(copia);
    exit(0);
  } else {
    while(wait(NULL)!=pid) {};
  }
// Punto F
  exit(0);
}
```

Considerando que:
- Las llamadas al sistema y el resto de funciones de librería que se incluyen en el programa nunca fallan.
- Hay suficientes marcos de página libres en el sistema.
- Todas las funciones externas se incluyen en una única biblioteca cuyo texto y datos ocupa 3,7 KB. Dicha biblioteca se enlazó de forma dinámica al crear el ejecutable.
- El texto (código) del programa tras la compilación ocupa 2,5KB. Incluye el código para la carga dinámica de bibliotecas.
- El contenido inicial de la pila al lanzar a ejecución el programa ocupa tan solo 32 bytes.
- Un entero en C ocupa 4 bytes.

a) Identifique qué regiones lógicas (y su tamaño en páginas) constituyen la imagen de memoria del proceso o procesos que se crean al lanzar a ejecución dicho programa en los puntos marcados en el código como A, B, C, D, E y F.



b) ¿Cuántos fallos de página se producirán al pasar del punto B al punto C?. Y del punto C al D?

C → D 1 fallo [acceso a la variable 'copia']

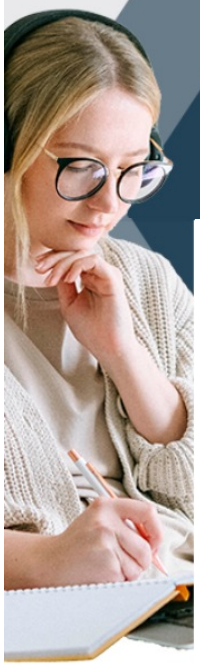
D → E 1 fallo [escritura del heap]

Escaneado con CamScanner

¡Adiós a las imperfecciones de la piel con My Clarins! Habla con el experto en CLARINS.COM

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

CURSOS INTENSIVOS PARA EXÁMENES DE
CONVOCATORIA ORDINARIA
Y EXTRAORDINARIA



Academia especializada en estudios
de la Facultad de Informática
Maths informática

Cuestión 4. (1.5 puntos)

Considere el siguiente código que representa el comportamiento de un hilo lector y otro escritor:

```
int a=0; /* shared variable */

void reader(){
 while(1){
   ReaderEnter();
   printf("a");
   ReaderExit();
   /* Sleep for some time */
   usleep(...);
  }
}
```

```
void writer(){
 while(1){
   WriterEnter();
   a=a+3;
   WriterExit();
   /* Sleep for some time */
   usleep(...);
 }
}
```

a) **Suponiendo que en el programa concurrente hay un número indeterminado de hilos lectores y escritores, implemente las funciones ReaderEnter(), ReaderExit(), WriterEnter() y WriterExit() para satisfacer las restricciones del problema de los lectores-escritores. Utilice para ello cerrojos, variables condicionales y otras variables compartidas.**

```
int nr.readers = 0,   lectores dentro de la seccion critica o que pueden estar
int nr.writers = 0,   escritor dentro de la seccion critica
mutex mutex.
condvar wrcond.
condvar rdcond

void ReaderEnter() {
    lock(mutex)
    nr.readers ++.
    while(nr.writers >0)
        cond-wait(rdcond, mutex).
    unlock(mutex).
}

void ReaderExit() {
    lock(mutex).
    nr.readers --.
    if(nr.readers ==0)
        cond-signal(wrcond).
    unlock(mutex)
}
```

```
void WriterEnter() {
    lock(mutex).
    while(nr.writers >0 || nr.readers >0)
        cond-wait(wrcond, mutex).
    nr.writers ++.
    unlock(mutex).
}

void WriterExit() {
    lock(mutex).
    nr.writers --.
    if(nr.readers == 0) cond-signal(wrcond).
    else cond-signal(rdcond).
    unlock(mutex)
}
```

b) **Para la solución proporcionada en el apartado anterior, indique si hay algún tipo de hilo (lector o escritor) al que se de preferencia sobre el otro en el acceso a la sección crítica. Justifique su respuesta.**

tienen prioridad los lectores, ya que en el caso de que haya un lector dentro de su seccion critica, los escritores no pueden entrar en su seccion critica pero si nuevos lectores

→ Solucion con semaforos y con la misma prioridad lec/esc

WUOLAH

Escaneado con CamScanner

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Cuestión 5 (1.5 puntos)

Un sistema UNIX dispone de un disco con una partición de 256KiB formateada con un sistema de ficheros tipo FAT, con bloques de 4KiB, montada en el directorio `/mnt/data`. El sistema mantiene una lista de bloques libres, gestionada como un fichero oculto llamado `bloques_libres` en el directorio raíz (marcado como tipo BL). Cuando se borra o trunca un fichero, los bloques que se liberan se añaden a la lista de bloques libres por el comienzo de la misma (nota: los bloques que se liberaran se encuentran enlazados en la lista de bloques libres manteniendo el orden que tuviesen en el fichero borrado o truncado). Cuando el sistema tiene que asignar un bloque lo coge siempre del comienzo de la lista de bloques libres. En un momento dado el sistema de ficheros tiene únicamente un fichero en su directorio raíz, llamado `datos.dat`. En las tablas se muestra el contenido de la FAT y el directorio en ese momento.

FAT

Entrada	Id Bloque	Entrad	Id Bloque
0	2	8	<EOF>
1	6	9	10
2	1	10	12
3	5	11	14
4	<EOF>	12	13
5	4	13	8
6	7	14	15
7	9	15	0

DIRECTORIO

Nombre	Tipo	Tamaño (B)	Bloque
bloques_libres	BL	53248	11
datos.dat	F	9216	3

Refleje en las tablas de abajo el efecto sobre las estructuras del Sistema de Ficheros de:

a) La ejecución del siguiente fragmento de código por un proceso:

```
int fd; char buf[10240];
... se asigna valor a buf ...
fd = open("/mnt/data/items.dat", O_WRONLY|O_CREAT|O_TRUNC, 0660);
write(fd, buf, sizeof(buf));
```

b) La ejecución de la siguiente orden en el shell:

```
cp /mnt/data/datos.dat /mnt/data/datos.bak
```

c) La ejecución del siguiente fragmento de código c:

```
int fd1, fd2; char buf1;
fd1 = open("/mnt/data/data.dat", O_WRONLY);
lseek(fd1, 4096, SEEK_SET);
fd2 = open("/mnt/data/items.dat", O_RDONLY);
while (read(fd2, &buf1, 1))
    write(fd1, &buf1, 1);
```

FAT

Entrada	Id Bloque	Entrad	Id Bloque
0	2	8	< EOF >
1	< EOF >	9	10
2	1	10	12
3	5	11	14
4	6	12	13
5	4	13	8
6	< EOF >	14	15
7	9	15	< EOF >

DIRECTORIO

Nombre	Tipo	Tamaño (B)	Bloque
bloques_libres	BL	28672	7
datos.dat	F	14336	3
items.dat	F	10240	11
datos.bak	F	9216	0

Reservados todos los derechos.
No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

Escaneado con CamScanner

¡Adiós a las imperfecciones de la piel con My Clarins! Habla con el experto en CLARINS.COM

SISTEMAS OPERATIVOS - PROBLEMAS
14 de Enero de 2019

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Problema (2.5 puntos) Problema de Ficheros/Procesos/Threads

Supóngase un sistema de ficheros tipo UNIX con las siguientes características:

- i-nodos de 48 bytes (para descripción y atributos del fichero) más 10 punteros directos, un puntero indirecto simple y un puntero indirecto doble.
- Tamaño de los punteros a bloques y a i-nodos: 32 bits.
- Los directorios tienen entradas con nombres de tamaño fijo de 100 caracteres
- Tamaño de bloques de 4 KiB

1. Cuando un usuario lista el contenido de su directorio "/home/usuarioso" (utilizando el comando "ls /home/usuarioso"), ¿cuántos bloques de disco se leen -en el caso mejor- asumiendo que no hay ningún dato relacionado con el sistema de ficheros en memoria RAM? Justifique la respuesta.

2. Indique el contenido y cuantos bytes son necesarios para almacenar el directorio cuyo listado con el comando ls -li se muestra a continuación:

i-node	flags	enlaces	usuario	grupo	tamaño	fecha	nombre
358304	-rwxrwxr-x	1	usuarioso	usuarioso	8448	dic 3 10:37	ejemploest
358275	-rw-rw-r--	1	usuarioso	usuarioso	166	dic 3 10:33	ejemploest.c
358295	-rwxrwxr-x	2	usuarioso	usuarioso	8432	dic 2 19:27	ejemplomem
358292	-rw-rw-r--	1	usuarioso	usuarioso	169	dic 2 19:27	ejemplomem.c
358302	-rw-rw-r--	1	usuarioso	usuarioso	1161	dic 2 18:44	ejercicio.c
271621	-rwxrw-rw-	1	usuarioso	usuarioso	176993	sep 27 11:38	Problems.pdf
358311	-rw-rw-r--	1	usuarioso	usuarioso	2736	dic 3 10:33	libestatica.a
358308	-rw-rw-r--	1	usuarioso	usuarioso	46	dic 3 10:32	miinclude.h
358295	-rwxrwxr-x	2	usuarioso	usuarioso	8432	dic 2 19:27	proceso1
309903	-rw-rw-r--	1	usuarioso	usuarioso	77	dic 3 10:19	suma.c

206660

3. A continuación se ejecutan los siguientes comandos:
   cp Problems.pdf Problems2.pdf
   ln –s ejemploest example

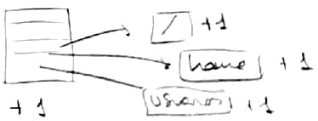
   ¿qué estructuras del sistema de ficheros (mapa de bits, inodos, directorios) se modifican y crean como consecuencia de la ejecución de cada comando?

4. ¿Con la información disponible, completa la información que disponemos del contenido del inodo del fichero ejemplomem? Los campos cuyo contenido no se conozca se pueden dejar en blanco o suponer un contenido razonable.

1 /home/usuarioso

leemos el i-nodo de cada directorio desde disco para averiguar cual es su bloque. leemos el bloque de datos para encontrar el numero de i-nodo del subdirectorio (os con /, home, usuarioso)

Hay 3 accesos para leer i-nodos y 3 accesos para leer bloques de datos. En caso mejor los 3 nodos estan en el mismo bloque y cada directorio ocupa 1 bloque, asi que traemos a memoria 4 bloques



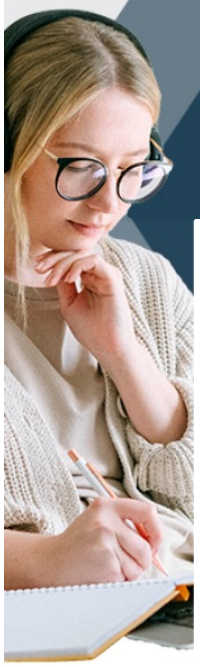
4 bloques.

Escaneado con CamScanner

¡Adiós a las imperfecciones de la piel con My Clarins! Habla con el experto en CLARINS.COM

Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.

CURSOS INTENSIVOS PARA EXÁMENES DE
CONVOCATORIA ORDINARIA
Y EXTRAORDINARIA



2 para almacenar el directorio de esa lista

(n de filas + . + ..) * (4 de inodo + 100 de nombre)B =
= 12 * 104 = 1248 Bytes

el contenido de cada fila:

nombre del fichero o directorio y número de inodo

3 $ cp Problems.pdf Problems2.pdf

crea el fichero Problems2.pdf donde copia el contenido de Problems.pdf (tamaño 176993 Bytes) Va a ocupar 44 bloques, aunque reservamos 44 bloques en el mapa de bits (+1 enlace indirecto), crear un inodo nuevo y rellenarlo con el contenido del fichero y añadir una entrada de directorio con el nombre Problems2.pdf y el número inodo

$ ln -s ejemplotest example

crea un enlace simbólico para ello se crea un inodo nuevo con los datos del enlace y una entrada de directorio con el nombre example y el número de inodo Este nuevo inodo contiene la información de la ruta hacia el fichero ejemplotest (origen) = '/home/usuario/ejemplotest'

4 información del inodo del fichero - ejemplonuevo

Tamaño: 8432 enlaces: 2
Permisos: 0775 (-rwxrwxr-x)
Modificación: 2019-12-2 19:27

ocupa 3 bloques de datos → ocupa los 3 primeros enlaces directos
los 7 directos + 1 indirecto simple + 1 indirecto doble
serán NULL

Academia especializada en estudios de la Facultad de Informática
Maths informática
Reservados todos los derechos. No se permite la explotación económica ni la transformación de esta obra. Queda permitida la impresión en su totalidad.
WUOLAH
Escaneado con CamScanner