SISTEMAS OPERATIVOS - CUESTIONES
3 de julio de 2018

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Cuestión 1. (1 punto) Un dispositivo de memoria flash de 1 MiB de capacidad, contiene un sistema de ficheros FAT con bloques de 1KiB y direcciones de disco de 16 bits. El sistema de ficheros está montado en Linux usando /mnt como punto de montaje. En dicho dispositivo se crea un fichero `/mnt/test.txt` y se escriben 10 bytes. Después se adelanta 5000 bytes el puntero de posición del fichero y se escriben otros 100 bytes. Se pide:

a. ¿Cuántos bytes son necesarios para almacenar la tabla FAT?

b. ¿Cuántos bloques ocupará el fichero `/mnt/test.txt` en disco?

c. Indique si los comandos que se indican a continuación funcionarán correctamente o no, indicando qué hace cada comando. Justifique la respuesta. (Suponer que el directorio /tmp existe y que las rutas que se pasan como último parámetro de los comandos no corresponden a ningún fichero existente.)

```
$ ln /mnt/test.txt /mnt/testA
$ ln -s /mnt/test.txt /tmp/testB
```

Cuestión 2. (1 punto) Responda brevemente a las siguientes preguntas: (1) ¿En qué directorio se alojan por convenio los ficheros de dispositivo en un sistema GNU/Linux? (2) Si se lista el directorio en cuestión con la orden `ls -l` tal como aparece más abajo, ¿qué representan los dos números separados por coma? (3) Si se desea instalar un módulo en el kernel con su código implementado en `driver-leds.c`, y asociarlo a un fichero de dispositivo tipo carácter llamado `file-leds`, ¿qué comandos hay que utilizar y en qué orden? Se supone que el fichero `Makefile` está disponible.

```
$ ls -l
...
brw-r----- 1 root disk 3, 0 Nov 19 10:20 hda
brw-r----- 1 root disk 3, 1 Nov 19 10:20 hda1
crw-rw---- 1 root uucp 4, 64 Nov 19 10:20 ttyS0
crw-rw-rw- 1 root root 1, 8 Nov 19 10:20 random
...
```

(1)

(2)

(3)

Cuestión 3. (1 punto) En un sistema tipo UNIX, un proceso ejecuta el siguiente programa, que pretende obtener la suma de los elementos de un vector de enteros utilizando procesos e hilos:

```c
#include <stdio.h>
#include <limits.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <pthread.h>

/*  La función lee un vector del fichero cuya ruta
se pasa como parámetro. Devuelve un puntero al
vector y el # de elementos.
*/
int* lee_vector(char* fichero, int* n_elementos){ …
}

struct arg_struct {
       int arg1;
       int arg2;
       int *arg3;
};

/* Almacena el sumatorio del vector
   (args->arg3) en el rango
   [args->arg1:args->arg2] en la variable
   suma_pthread */
int suma_pthread=0;
void* sumatorio_pthread(void* arg)
{
    int i;
   struct arg_struct *args =
      (struct arg_struct *)arg;

   int start_idx = args->arg1;
   int end_idx = args->arg2;
   int *vector = args->arg3;


   for (i=start_idx;i<end_idx;i++)
      suma_pthread+=vector[i];
}

/* Devuelve el sumatorio del vector
   en el rango [start_idx:end_idx]*/
int sumatorio(int* vector,int start_idx,
   int end_idx) {
   int i, suma=0;

   pthread_t thread;
   int n_elems  = end_idx-start_idx;
   int half_idx = start_idx+n_elems/2;

   /* Creación de rangos para el pthread */
   struct arg_struct args;

   args.arg1 = start_idx;
   args.arg2 = half_idx;
   args.arg3 = vector;
   pthread_create(&thread, NULL,
      sumatorio_pthread, (void *)&args);

   for (i=half_idx;i<end_idx;i++)
      suma+=vector[i];

   pthread_join(thread, NULL);

   return suma+suma_pthread;
}

int main(int argc, char *argv[]) {
   int n_elem;
   int* vector;
   pid_t pid;
   int suma1=0,suma2=0;

   if (argc!=2) {
      fprintf(stderr,
         "Uso: %s <fichero>\n",argv[0]);
      exit(1);
   }

   vector=lee_vector(argv[1],&n_elem);

   if ((pid=fork())==0) {
      /* Proceso hijo */
      suma1=sumatorio(vector,0,n_elem/2);
      exit(0);
   } else {
      /* Proceso padre */
      suma2=sumatorio(vector,
         n_elem/2,n_elem);
   } else {
      fprintf(stderr,"Error en fork()\n");
      exit(2);
   }

   /* Esperar al proceso hijo */
   while(wait(NULL)!=pid) {}
   printf("El sumatorio del vector es %d\n",
      suma1+suma2);

   free(vector);

   return (0);
}
```

Responda a las siguientes cuestiones:

a) ¿Cuántos hilos pueden ejecutarse simultáneamente como máximo? ¿Y cuántos procesos? Razonar la respuesta

b) ¿El programa anterior está codificado correctamente para obtener la suma de todos los elementos del vector? Si no fuera así indicar el motivo y una posible solución.

Cuestión 4. (1 punto) Indicar de forma justificada cuáles de las siguientes transiciones entre los estados de un proceso no se pueden producir en un sistema con una política de planificación SJF (*Shortest Job First*) no expropiativa:

1) Bloqueado a listo.
2) Ejecutando a listo.
3) Ejecutando a bloqueado.
4) Listo a bloqueado.
5) Listo a ejecutando.

C**uestión 5. (1 punto)** Suponer que los dos procesos siguientes, p1 y p2, se ejecutan concurrentemente y comparten los semáforos s1 y s2 (cada uno inicializado a 1) y la variable entera x (inicializada a 0)

```
void p1() {
  do {
    wait(s2);
    wait(s1);
    x++;
    signal(s2);
    signal(s1);
  } while (1);
}
```

```
void p2() {
  do {
    wait(s1);
    wait(s2);
    x--;
    signal(s2);
    signal(s1);
  } while (1);
}
```

a. ¿Puede la ejecución concurrente acabar en interbloqueo? En caso afirmativo, proporcionar una secuencia de ejecución en la que se produzca el interbloqueo.
b. ¿Puede la ejecución concurrente acabar en la espera indefinida de uno de ellos? En caso afirmativo, proporcionar una secuencia de ejecución en la que se produzca la espera.

(a)

(b)

Cuestión 6. (1 punto) En una arquitectura que usa direcciones virtuales de 8 bits, la imagen de memoria de los procesos consta de 8 páginas. Un proceso tiene los siguientes datos en su Tabla de Páginas (TP):

Pág.	Marco	Validez	Protección	Modificada
0	0		RW-	
1	0		--X	
2	1	✔	RW-	
3	3	✔	--X	

Pág.	Marco	Validez	Protección	Modificada
4	2		RW-	
5	0		RW-	
6	3		RW-	✔
7	2		RW-	

NOTA: El proceso tiene asignados los marcos 0-3

a. Indique el tamaño de las páginas en bytes.

b. Teniendo en cuenta que el código del proceso está en la página 0 y que los datos en las páginas 1-4, indique razonadamente si se produce acierto, fallo o violación de protección o segmento, así como las posibles modificaciones en la TP, al generar las siguientes direcciones virtuales
   1. Escritura en la dirección 0x5C
   2. Lectura de la dirección 0x30
   3. Ejecución del contenido de la dirección 0x00

SISTEMAS OPERATIVOS - PROBLEMAS
3 de julio de 2018

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Problema 1. (2 puntos) Se ha diseñado un sistema operativo que organiza la información en el disco duro de 4GiB mediante nodos-i. Éstos están organizados con 2 índices directos, 1 índice indirecto y un índice con doble indirección, para bloques de 32B. El sistema operativo gestiona un disco duro que en un momento dado tiene la siguiente información:

i-nodo	2	4	7	8	9	10	11	12				
enlaces	4	3	2	1	1	1	1	2				
Tipo	D						F					
Directo		2	4	5	10	15	9					
Directo	null	null	null	6	16	8	null	null				
Indirecto	null	null	null	7	null	null	null	null				
Indirecto doble	null	null	null	null	null	null	null	null				

Tabla de nodos-i

1		2		3		4		7
.	2	.		.	12	.		18
..	2	..	2	..		..		19
home	7	Apuntes	12	Tema1.odt	8			20
usr	4	tmp.txt	11	Practica2.txt	9			
				Solucion.odt	10			

Lista de bloques relevantes (los bloques que no aparecen aquí contienen datos sin formato conocido para el SO o están libres)

a. Rellena el mapa de bits de bloques de datos (hasta el bloque 20), donde 0 es libre y 1 ocupado, el primer índice se corresponde con el bloque 0.
b. Rellena los datos que faltan en la tabla de nodos-i y en la lista de bloques relevantes
c. Indicad el tamaño máximo de fichero posible con esta configuración
d. Considere el siguiente código, indique qué llamadas al sistema modificarán seguro las tablas anteriores, describa cómo se quedaría la Tabla de nodos-i y por qué.

```
...
char buf[32] = "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA\n";
int fd1 = open("/home/tmp.txt", O_RDWR);
int fd2 = open("/usr/copia.odt", O_CREATE);
int fd3 = open("/home/Apuntes/Solucion.odt", O_RDWR);
lseek(fd1, 2304, SEEK_SET);
write(fd1, buf,32);
while((c=getc(fd3))!=EOF) putc(c,fd2);
...
```

Problema 2. (2 puntos) En la cafetería de una Facultad se sirven bocadillos de dos tipos: jamón (0) y tortilla (1). Cuando un estudiante desea comer un bocadillo de un determinado tipo ha de ir a recogerlo a la barra. Si en ese instante quedan bocadillos del tipo elegido, se servirá uno y comerá. En otro caso, el estudiante deberá dar aviso al cocinero, que estará bloqueado en ese momento. Al ser despertado, se mirará qué tipo de bocadillos hay que reponer, repondrá N bocadillos de todos los tipos que falten y volverá a bloquearse. Entonces el estudiante podrá servirse el bocadillo y comer.

Un número arbitrario de estudiantes y el cocinero (hilos del programa concurrente) se comportan del siguiente modo:

```
void Estudiante(int tipoBocadillo){
    while (true) {
       conseguirBocadillo(tipoBocadillo);
       comer();
    }
}
```

```
void Cocinero(){
    while (true) {
       servirBocadillos();
    }
}
```

Implemente las funciones `conseguirBocadillo(int)` y `servirBocadillos()` empleando variables compartidas y los mecanismos de comunicación y sincronización que se consideren necesarios. **Nota:** Además de proporcionar el código de la solución, se ha de describir claramente para qué sirve cada variable/recurso de sincronización utilizado, así como su valor inicial.