UCM - Facultad de Informática

Sistemas Operativos - Ingeniería Informática

22 de junio de 2017

Apellidos: _____ **Grupo:** _____

Nombre: _____ **NIF:** _____

Cuestiones

C1.- (1pt) Un dispositivo de memoria flash de 64 MiB de capacidad y bloques de 1KiB, contiene un sistema de ficheros FAT. Conteste a las siguientes preguntas:

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

b) Si llenamos la partición con ficheros de 500B, ¿qué porcentaje de disco se desperdiciaría?

c) Si quisiésemos reducir el porcentaje de disco desperdiciado ¿qué cambio podríamos hacer en el tamaño de bloque? Ponga un ejemplo de cómo afectaría el cambio a la tabla FAT y del nuevo porcentaje de disco desperdiciado.

NOTA: Asuma que todos los bloques de datos se asignan a datos de los ficheros.

C2.- (1pt) Explique qué es lo que hace el siguiente programa e indique si puede haber variaciones en función del orden de planificación de los procesos. Indique el contenido final de los ficheros y los mensajes que se mostrarán por terminal. **NOTA:** Recordad que la función `lseek()` devuelve la nueva posición del puntero de posición (bytes desde el comienzo del fichero).

```
1  int child_swap=0;
2
3  int main(void){
4    int fd1,fd2,i,pos;
5    char c;
6
7    fd1 = open("outfile.txt", O_CREAT | O_TRUNC
8               | O_RDWR, S_IRUSR | S_IWUSR);
9    for (i=0; i < 4; i++) {
10     write(fd1, "AAAA", 4);
11     pos = lseek(fd1, 0, SEEK_CUR);
12
13     if (fork() == 0) { // Hijo
14
15       fd2 = open("outfile.txt", O_WRONLY);
16       lseek(fd2, pos, SEEK_SET);
17
18       if (child_swap == 0) {
19         write(fd2, "BBBB", 4);
20         child_swap = 1;
21       } else {
22         write(fd2, "CCCC", 4);
23         child_swap = 0;
24       }
25       close(fd2);
26       exit(0);
27
28     } else {  // Padre
29       wait(NULL);
30     }
31   }
32   lseek(fd1, 0, SEEK_SET);
33   printf("File content is:\n");
34   while (read(fd1, &c, 1) > 0)
35   printf(" %c", (char) c);
36   printf("\n");
37   close(fd1);
38   exit(0);
39 }
```

C3.- (1pt) Indique cuáles de las siguientes afirmaciones son verdaderas y cuáles falsas. En el caso de que considere que la respuesta es falsa, indique el motivo.

a) Cuando un módulo del kernel no está en uso (contador de referencia igual a 0), el kernel descarga el módulo automáticamente.

b) Un proceso de usuario sólo puede realizar operaciones de Entrada/Salida solicitándolas a través de una interrupción software (`trap`).

c) Cuando un módulo del kernel termina, basta con que éste haga `exit()` para descargarse.

d) Si un módulo del kernel realiza una operación no permitida, como por ejemplo acceso a una región de memoria no disponible, el kernel lo detecta y descarga el módulo.

C4.- (1pt) Contestar a las siguientes preguntas sobre diferentes formas de planificar la ejecución de este conjunto de tareas.

Tarea	Llegada	CPU	E/S	CPU
T1	0	3	2	2
T2	2	3	1	2
T3	1	1	3	3
T4	0	5	1	

a) ¿Cuál será la última tarea en completar su ejecución si se aplica **SJF** expropiativo?
b) ¿Cuál será el tiempo total de ejecución con **FCFS**?

NOTA: marcar el tiempo de CPU coloreando y el de E/S rayando

SJF expropiativo

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
T1																								
T2																								
T3																								
T4																								

FCFS

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
T1																								
T2																								
T3																								
T4																								

C5.- (1pt) Dada la siguiente secuencia de referencias a direcciones de memoria virtual generadas por un sólo programa en un sistema con paginación pura:

`0x1A0 0x1B8 0x200 0x420 0x40C 0x3F8 0x204 0x322 0x62C 0x13A 0x380 0x5D2 0x202`

a) Obtenga la cadena de referencias (secuencia de números de página generadas por el programa), suponiendo un tamaño de página de 256 Bytes.
b) Determine razonadamente el número de fallos de página usando como estrategia de sustitución el algoritmo segunda oportunidad (**Reloj**), suponiendo que hay cuatro marcos de página disponibles para el programa y que están inicialmente vacíos.

REF.	1A0	1B8	200	420	40C	3F8	204	322	62C	13A	380	5D2	202
pag													
m0													
m1													
m2													
m3													

C6.- (1pt) Dado el siguiente código que simula el problema del **Productor/Consumidor** extiéndalo para que funcione para múltiples **Productores/Consumidores**:

```
#define MAX_BUF 1024 // tamanio del buffer
sem_t elementos;     // Inicializado a 0
sem_t huecos;        // Inicializado a MAX_BUF
int buffer[MAX_BUF]; // buffer comun
int cons, prod = 0;  // Punteros prod/cons al buffer
```

```
1  void Productor(void){
2    int dato;
3    while(1){
4      dato = producir_dato();
5      // un hueco menos
6      sem_wait(&huecos);
7      buffer[prod] = dato;
8      prod = (prod + 1) % MAX_BUF;
9      // un elemento mas
10     sem_post(&elementos);
11   }
12 }
13
```

```
14 void Consumidor(void){
15   int dato;
16   while(1){
17     // un elemento menos
18     sem_wait(&elementos);
19     dato = buffer[cons];
20     cons= (cons+ 1) % MAX_BUF;
21     // un hueco mas
22     sem_post(&huecos);
23     consumir_dato(dato);
24   }
25 }
```

Problemas

P1.- (2 pts) Un sistema de ficheros de un SO diseñado a partir de UNIX utiliza bloques de disco de 32 bytes de capacidad. Para el direccionamiento de estos bloques se utilizan punteros de 16 bits. Cada nodo-i tiene 1 puntero de direccionamiento directo, 1 puntero indirecto simple y 1 puntero indirecto doble. Parte del contenido del sistema de ficheros está indicado en las siguientes tablas:

Mapa de bits de bloques de datos. El primer índice se corresponde con el bloque 0 y '1' indica ocupado:

```
Mapa de bits: 0001 0111 0100 1010 0000 ..... 0
```

nodo-i	2	3	4	6	7	8	9
Enlaces	NA	NA	NA	NA	1	1	1
Tipo F/D	D	D	D	D	F	F	F
Directo	3	5	6	7	9	12	14
Ind. Simple	-	-	-	-	-	-	-
Ind. Doble	-	-	-	-	-	-	-

Bloque 3

.	2
..	2
var	3
tmp	4

Bloque 5

.	3
..	2
log	6

Bloque 6

.	4
..	2
tar	8

Bloque 7

.	6
..	3
auth.log	9
dmesg	7

Bloque 9

AMD AuthenticAMD

1. (0.25 ptos) Indique el tamaño máximo de los ficheros.
2. (0.5 ptos) Dibuje el árbol de directorios empleando óvalos para los directorios y rectángulos para los ficheros (la raíz del árbol es el nodo-i 2)
3. (0.5 ptos) Indique la modificación del sistema de ficheros (tablas de nodos-i, lista de bloques y mapa de bits) producida tras el arranque del sistema en el que se escribe en el en fichero de log `auth.log` la frase `Linux version 4.4.0-78-generic (4.4.0-78.99 14.04.2)`[1] suponiendo que cada carácter ocupa un byte.
4. (0.75 ptos) Un usuario con permisos sudo crea un enlace físico llamado `enlace_fisico` al fichero `auth.log` y uno lógico llamado `enlace_logico` en el directorio `tmp`. Responded:
   (a) ¿Que comandos debe de teclear dicho usuario?
   (b) ¿Como se modifica el sistema de ficheros?

[1] `Linux version 4.4.0-78-generic (4.4.0-78.99 14.04.2)` contiene 53 caracteres

P2.- (2 pts) Problema del Puente Viejo. Un puente cruza un río de este a oeste, como se trata de un puente muy viejo, sólo dispone de un carril y sólo puede haber, como mucho, tres coches de manera simultánea sobre él. Para simular el comportamiento ejecutaremos múltiples hijos que ejecuten el siguiente código:

```
typedef enum Direccion {ESTE_OESTE=0, OESTE_ESTE=1};

void cruzar(Direccion mi_direccion){
    entrar_puente(mi_direccion);
    sleep(rand()%10+1);
    salir_puente(mi_direccion);
}
```

Implemente las funciones `entrar_puente` y `salir_puente` teniendo en cuenta que:

- Si llega un coche al puente y este está vacío, deberá de cruzar inmediatamente.
- Si llega un coche y encuentra el puente ocupado por otros coches cruzando en dirección contraria, deberá de esperar a que el puente esté vacío.
- Si llega un coche y encuentra que hay menos de tres coches cruzando en su mismo sentido, deberá de cruzar inmediatamente.
- Si llega un coche y encuentra que hay tres coches cruzando en su mismo sentido, deberá de esperar a que el primero de ellos abandone el puente para cruzar.

La implementación deberá de usar mutex y variables de condición como único mecanismo de sincronización así como las variables globales que estime oportunas. Comente la posibilidad de inanición de la solución propuesta poniendo un ejemplo de situación en la que pudiese darse si la hubiese.