SISTEMAS OPERATIVOS - CUESTIONES
3 de Febrero de 2014

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Cuestión 1. (0,75 puntos) Un dispositivo de memoria flash de 1 MB de capacidad y bloques de 1KB, contiene un sistema de ficheros FAT. Las direcciones son de 16 bits. En dicho dispositivo se crea un fichero y se escriben 10 bytes. Después se adelanta 5000 bytes el apuntador 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 de datos tendrá el fichero?
c. Si el fichero se cierra y en un momento posterior se quieren leer los 100 últimos bytes calcular el número de accesos al dispositivo necesarios (para el caso peor y para el caso mejor). Asumir que no hay ningún dato relacionado con el sistema de ficheros en memoria *RAM,* que el fichero se encuentra en el directorio actual y que el directorio cabe en un bloque.

Cuestión 2. (0,75 puntos) Describe brevemente para qué sirven los dos números denominados `major` y `minor`. ¿Qué hace la función register_chrdev?

```
int register_chrdev(unsigned int major, const char* name,
                    struct file_operations* fops);
```

Describe qué son los parámetros que recibe y qué es el número que retorna.

Cuestión 3. (0,75 puntos) Supongamos un módulo que implementa la siguiente función para un dispositivo tipo carácter, y que dicha función se configura como operación de lectura del driver del dispositivo:

```
static ssize_t device_read(struct file *filp, char *buffer,
                           size_t length, loff_t * offset);
```

Supongamos que el módulo se compila y se instala, y que se crea un archivo de dispositivo con este módulo (usando la orden `mknod`). Poner un ejemplo de orden Linux en que se ejecutaría automáticamente esta función. Responde brevemente a las siguientes cuestiones: (a) ¿Qué debe almacenar la cadena buffer? (b) ¿Qué sentido tiene utilizar la función `put_user(*(msg_ptr++), buffer++)` dentro de device_read?

Cuestión 4. (0,75 puntos) Considere el siguiente código:

```
int a =0;
int main() {
 pid_t pid;
 int b = 0;
 int i ,ret;

 for (i=0;i<3;i++) {
  if ( (pid=fork()) == 0) {
       a = a+2;
       b = b+2;
       ret = execlp ("echo", "echo", "Proceso", (char *)0);
       printf("Child. pid=%u ppid=%u. i=%d a=%d b=%d
      \n",getpid(),getppid(),i,a,b);
```

```
    }
    else {
        wait(NULL);
        a++; b++;
        printf("Parent. pid=%u ppid=%u. i=%d a=%d b=%d
       \n",getpid(),getppid(),i,a,b);
    }
  }
}
```

Asumiendo que el proceso original tiene PID 100 (y es hijo de *Init*) y que los PID se van asignando de forma consecutiva, indique qué se mostrará por pantalla al ejecutar el código.

Cuestión 5. (0,75 puntos) Considere el siguiente código:

```
char buf1[5] = "XXXX";
char buf2[5] = "AAAA";
int main() {
    int fd1,fd2;
    fd1=open("prueba",O_RDWR | O_CREAT | O_TRUNC, 0666 );

    if ( fork() == 0) {
      fd2=open("prueba",O_RDWR );
      write(fd1,buf1,4);
      write(fd2,buf2,4);
      lseek(fd1,0,SEEK_SET);
      read(fd1,buf1,4);
      close(fd2);
      close(fd1);
    }
    else {
      wait(NULL);
      write(fd1,buf2,4);
      read(fd1,buf1,4);
      close(fd1);
    }
}
```

Indique si la ejecución del código generará algún error. En cualquier caso, ¿cuál será el contenido del fichero *prueba* al finalizar la ejecución? ¿Qué contendrá la variable "buf1" tanto al finalizar el proceso hijo como al finalizar el padre?. **Justifique la respuesta**.

Cuestión 6. (0,75 puntos) Los requisitos que debe ofrecer cualquier solución para resolver adecuadamente el problema de la sección crítica son: exclusión mutua, progreso y espera limitada. Explique brevemente en qué consiste cada uno de los tres.

Cuestión 7. (0,75 puntos) En un sistema con memoria virtual paginada indicar cómo funciona el algoritmo del reloj (información que almacena para cada marco de página, número de fallos de página,…) suponiendo una memoria física de cuatro marcos de página (inicialmente los marcos de página están vacíos) y usando como ejemplo la siguiente cadena de referencias: a d c a b d f a e f a e d f a g

Cuestión 8. (0,75 puntos) En un sistema con memoria virtual paginada indicar qué sucede y cuáles de las acciones son realizadas por el sistema operativo (especificando a qué estructuras de datos accede y cómo las modifica) en los siguiente casos: a) un proceso intenta escribir en una página de sólo lectura; b) un proceso intenta acceder a una dirección virtual de su espacio de direcciones correspondiente a una página que no está en memoria.

SISTEMAS OPERATIVOS - PROBLEMAS
3 de Febrero de 2014

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Problema 1. (2 puntos) El sistema de ficheros de un SO diseñado a partir de UNIX utiliza bloques de disco de 4096 bytes de capacidad. Para el direccionamiento de estos bloques se utilizan punteros de 32 bits. Cada nodo-i tiene 8 punteros de direccionamiento directo, 1 puntero indirecto simple y 1 puntero indirecto doble.

a. ¿Cuál será el tamaño máximo de un fichero suponiendo despreciable el espacio ocupado por el superbloque, el mapa de bits y la tabla de nodos-i? Si se modifica el tamaño de puntero pasándolo a 64 bits, ¿cuál será el nuevo tamaño máximo?
b. Supongamos que el sistema de ficheros contiene la información de las tablas de abajo (sólo se muestra el primer puntero directo). Dibuje el árbol del directorio empleando óvalos para los directorios y rectángulos para los ficheros. Complete el campo *Enlaces* en la tabla de nodos-i.

Tabla de nodos-i

nodo-i	1	2	3	4	6	7	9	10	12
Enlaces	NA	NA	NA	___	NA	NA	___	___	___
Tipo F/D	D	D	D	F	D	D	F	F	F
Directo	4	5	6	13	8	9	14	12	15

Lista de bloques:

Bloque 4

.	1
..	1
A1	2
A2	3
a3	4

Bloque 5

.	2
..	1
A5	6

Bloque 6

.	3
..	1
A6	7

Bloque 8

.	6
..	2
a8	9
a9	10

Bloque 9

.	7
..	3
a10	10
a11	12

Problema 2. **(2 puntos)** Para agilizar el paso por caja, cierto supermercado ha decidido implantar un sistema de cola único. Este sistema consiste en que los clientes que deseen pasar por caja esperen en una única cola, repartiéndose por las cajas libres en orden de llegada. El funcionamiento es el siguiente:

1. La caja libre se anuncia a través de un cartel luminoso.
2. Al llegar un nuevo cliente éste se coloca en la cola común a la espera de ser el primero y de que haya una caja libre anunciada en el cartel. Cuando esto ocurre, borra el cartel y se dirige a dicha caja, poniendo los artículos en la cinta.
3. En caso de que haya más de una caja libre, sólo un cajero deberá poner el número de su caja en el cartel, el resto de cajas libres esperaran a que el cartel esté vacío.

Escriba un programa que sincronice a los cajeros y a los clientes de acuerdo con la estructura siguiente:

```
// Variables compartidas
int cajaLibre=-1
```

```
thread_Caja(int n) {
 while(1) {
   //Espero a que el cartel esté vacío

   cajaLibre=n;
   //Avisar a un cliente

   atenderCaja(n);
 }
}
```

```
thread_Cliente(){
 // Esperar turno en la cola única

 // Guardar el número de la caja
 nCaja=cajaLibre;
 // Marcar el cartel como libre y
 // avisar

 ponerArtículos(nCaja);
 pthread_exit(NULL);
}
```

Nota: Asuma que en el proceso de compra, esto es una vez que un cajero está atendiendo mediante `atenderCaja(n)` y un cliente está poniendo artículos en la cinta correspondiente invocando `ponerArtículos(n)`, hay sincronización implícita entre ellos.