NOMBRE Y APELLIDOS: _____ NIA: _____

Examen de Sistemas Operativos - 2º Parcial
21 de Enero de 2010

NOTAS:

* Para la realización del presente examen se dispondrá de 2 horas.
* No se pueden utilizar libros ni apuntes, ni usar móvil (o similar).
* Responda cada pregunta en hojas distintas.

Teoría 1. ¿Qué es mejor, implementar un servidor que tiene mucho tráfico con procesos o con un pool de threads?

Explique en detalle su respuesta.

Teoría 2. Explique cómo se representa un archivo FAT y un archivo en UNIX con nodo-i. Suponga un bloque de 4 Kbyte y tamaño de la dirección de bloque de 4 bytes.

¿Cuántos accesos a la FAT y al nodo-i serían necesarios para acceder al byte 4096000?

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen de Sistemas Operativos - 2º Parcial
21 de Enero de 2010

Ejercicio 1. Considere un servicio de peluquería mixto para hombres y mujeres. Se pide controlar la entrada a la peluquería para un hombre y para una mujer, teniendo en cuenta las siguientes restricciones:

- No puede haber hombres y mujeres al mismo tiempo en la peluquería.
- La capacidad máxima de la peluquería es de 10 personas.
- Si alguien no puede entrar se va, es decir, no se queda esperando en la puerta. (Se considera que una persona no puede entrar cuando en la peluquería hay alguien del sexo opuesto o cuando está llena).
- Se deben utilizar semáforos para la sincronización de ambos procesos.
    - Las operaciones de semáforos a utilizar son wait y signal.
    - Se debe especificar el valor inicial de los semáforos usados.

A continuación, las dos funciones que tiene que completar el alumno en pseudocódigo: entrar_hombre / entrar_mujer representan el control que se debe hacer cuando un hombre / mujer pretende acceder a la peluquería.

```
entrar_hombre()
{
    /* no puede entrar un hombre
    si hay mujeres dentro o
    si está llena la peluquería */
}
```

```
entrar_mujer()
{
    /* no puede entrar una mujer
    si hay hombres dentro o
    si está llena la peluquería */
}
```

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen de Sistemas Operativos - 2º Parcial
21 de Enero de 2010

Ejercicio 2.

a) Se tiene un disco con sistema de ficheros Unix con las siguientes características:
   - Tamaño del bloque de 4 KBytes.
   - Direcciones de los bloques: 4 bytes.
   - Estructura del i-nodo:
     10 punteros directos.
     1 puntero indirecto simple.
     1 puntero indirecto doble.
     1 puntero indirecto triple.

Indica cuál es el número de bloques que ocupa un fichero de 10 MBytes, incluyendo tanto los bloques de datos como los de direcciones.

b) Se tiene un sistema de ficheros tipo Unix con la siguiente información:

Tabla de I-nodos:

Nº Inodo	2	3	4	5	
Tipo	Directorio	Directorio	Fichero	Enlace Simbólico	
Contador Enlaces Fis.	3	2	2	1	
Dirección Bloque Datos	11	12	13	14	
.					

Bloques de datos:

Nº Bloque	11	12	13	14	
Contenido	. 2	. 3	Datos del fichero	/d/f1	
	.. 2	.. 2			
	d 3	f1 4			
		f2 4			
		f3 5			

Indica cómo quedan los i-nodos y los bloques de datos después de realizar cada una de las siguientes operaciones (Hacer una tabla de i-nodos y de bloques de datos por cada apartado):

1- rm /d/f1
2- rm /d/f2
3- rm /d/f3
4- mkdir /d/d1

NOMBRE Y APELLIDOS: _____ NIA: _____

Examen de Sistemas Operativos - 2º Parcial
21 de Enero de 2010

Ejercicio 3. Se quiere controlar un máximo de 10 ascensores que se moverán para dirigirse a las plantas de destino. Se pide realizar la codificación de la función del thread ascensor (que simula el movimiento de un ascensor) y la del thread controlador (que permite atender las peticiones de usuarios de forma concurrente). Tenga en cuenta que:

- Un ascensor que se está moviendo para dirigirse a una planta destino no puede ser utilizado por el controlador.
- El ascensor debe poner un mensaje mostrando todas las plantas por las que pasa, en el momento en que está en ellas.
- Para el controlador, si se recibe una petición para un ascensor en movimiento, debe escribir un mensaje de Petición rechazada porque el ascensor N se está moviéndo. También debe mostrar un mensaje de error si se recibe una petición para un ascensor inexistente.
- Se proporciona la función `peticionparaAscensor` que cada vez que se ejecuta devuelve el ascensor a mover y la planta de destino.

```
#define MAXASCENSORES 10
int estancia[MAXASCENSORES];
int destino[MAXASCENSORES];
pthread_mutex_t mutexe[MAXASCENSORES];
pthread_mutex_t mutexd[MAXASCENSORES];
pthread_mutex_t mutexm[MAXASCENSORES];

void moveraotraplanta(){   sleep(1);   }

/* El parámetro indica el número de ascensor*/
void *ascensor(void *numasc){
    int asc;
    /* ... CODIGO DEL ALUMNO PARA ASIGNAR A asc EL ASCENSOR GESTIONADO POR ESTE
        THREAD ... */
    printf("Ascensor %d preparado\n", asc);
    while(1){
        //...........CODIGO DEL ALUMNO ...................
        printf("ascensor %d está en planta %d\n", asc, estancia[asc]);
        //...........CODIGO DEL ALUMNO ...................
    }
    pthread_exit(0);
}
void *controlador (void *numerodeAscensores){
    int asc,planta;
    printf("Preparado el control de ascensores\n");
    while(1){
        peticionparaAscensor(&asc, &planta);
        printf("Leida petición para que ascensor %d vaya a planta %d\n", asc,
                planta);
        //...........CODIGO DEL ALUMNO ...................
  }
}
```