SISTEMAS OPERATIVOS - CUESTIONES
12 de Febrero de 2016

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Cuestión 1. (0,75 puntos) Considere un sistema de ficheros FAT. Se dispone de 16KB para datos (esto es, la FAT y el directorio raíz van aparte) y los bloques son de 1KB. El contenido del directorio raíz se muestra a continuación:

Nombre	*Tipo*	*Fecha*	*Tamaño (bytes)*	*1er bloque*
tesis.tex	F. Regular	01/01/2016	5567	3
makefile	F. Regular	03/01/2016	1048	0
escudo.png	F. Regular	02/01/2016	4280	6

a) Escriba un posible contenido de la tabla FAT (indique el contenido de TODAS sus entradas)

b) Se quiere crear un enlace rígido a *tesis.tex*, y se intenta añadiendo esta entrada en el directorio raíz:

tesisLink.tex	F. Regular	01/01/2016	5567	3

Comente esta solución indicando si realmente es un enlace rígido. Justifique su respuesta.

Cuestión 2. (0,75 puntos) Un determinado módulo Linux se inicializa del siguiente modo:

```
dev_t start;
static struct
file_operations fops = {
  .read = my_read,
  .write = my_write,
  .open = my_open,
  .release = my_release
};
...
```

```
int init_module(void) {
  alloc_chrdev_region(&start, 0, 1, "disp");
  chardev=cdev_alloc();
  cdev_init(chardev,&fops);
  cdev_add(chardev,start,1);
  int major=MAJOR(start);
  int minor=MINOR(start);
  return 0;
}
```

Suponga que tras instalar el módulo major=250 y minor=0 se crea un fichero de dispositivo con el siguiente comando: mknod /dev/disp c 250 0. Explique el significado de cada uno de los parámetros de este comando. Indique cuál o cuáles de las funciones registradas por cdev_init se ejecutarán cuando se ejecute el comando: echo "hola mundo" > /dev/disp.

Cuestión 3. (0,75 puntos) 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) El programador de un driver en LINUX es el que especifica la interfaz de llamadas que se expone a los programas de usuario para interactuar con el dispositivo.
b) La función fopen() de la biblioteca estandar de C no se puede invocar desde un módulo del kernel.
c) Cuando un módulo del kernel no está en uso (contador de referencia igual a 0), el kernel descarga el módulo automáticamente.
d) Dos drivers en Linux podrían tener asignado el mismo major number, pero en tal caso gestionarían peticiones emitidas por programas de usuario (p.ej., lectura/escritura) sobre ficheros de dispositivo que tuvieran distintos minor numbers.
e) Un driver se compila como un programa normal, la única diferencia es que debería ejecutarlo un usuario con privilegios.

Cuestión 4. (0,75 puntos) Considere un sistema operativo que implementa semáforos generales (tipo sem_t) pero no así cerrojos ni variables condicionales. En este sistema, se desea construir un programa concurrente que precisa de barreras de sincronización. Implemente una barrera usando dos semáforos generales y otras variables compartidas . La barrera de sincronización, que se representará mediante el tipo de datos barrier_t (ya definido), se inicializará mediante la función barrier_init(). La operación de espera en la barrera se implementará mediante la función barrier_wait(). Por simplicidad en la implementación, suponga que ningún hilo del programa concurrente que acabe de retornar de barrier_wait(), volverá a invocar dicha función hasta que todos los demás hilos del programa hayan retornado de barrier_wait().

```
typedef struct {
   sem_t sem_mtx;
   sem_t sem_queue;
   int nr_max_threads;
   int threads_arrived;
} barrier_t;
```

```
void barrier_init(barrier_t* b,
                  int nthreads) {



}
```

```
void barrier_wait(barrier_t* b) {



}
```

Cuestión 5. (0,75 puntos) Un proceso en UNIX ejecuta el siguiente código:

```c
#include <stdlib.h>
#include <stdio.h>


#define CONSTANT 10

int num1 = CONSTANT;
int num2;

char string[] = "hello";

void* print_something(void*
arg){
   char* msg=(char*) arg;
   printf("Message: %s\n",msg);
   sleep(20);
   return NULL:
}
```

```c
int main(int argc, char *argv[]) {
 pthread_t thread;
 int *i=NULL;

 if ( (i=(int*)malloc((sizeof(int))) == NULL )
         return(-1);
 num2=argc;

 for(*i=0;*i<CONSTANT;*i=*i+1) {
     fprintf(stdout, "%s: %d, argc: %d\n", string, num1--, num2);
 }

 pthread_create(&thread,NULL,print_something,string);

 pthread_join(thread,NULL);

 return(0);
}
```

a) Para las siguientes variables/macros indique cuáles ocupan espacio en el ejecutable. Indique también la región del mapa de memoria en la que se encuentran.

Variable/Macro	Ocupa espacio en ejecutable (Sí/No)	Región del mapa de memoria
CONSTANT		
num1		
num2		
i		
*i		

b) Indique de qué regiones estará constituido el mapa de memoria del proceso cuando el hilo principal del programa se encuentre bloqueado en la llamada a pthread_join().

Cuestión 6. (0,75 puntos) Dada la siguiente secuencia de referencias a direcciones de memoria virtual generadas por un sólo programa en un sistema con paginación pura:

`0x3A0, 0x1B8, 0x200, 0x320, 0x40C, 0x6F8, 0x504, 0x322, 0x12C, 0x43A, 0x380, 0x6D2, 0x502`

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 LRU, suponiendo que hay cuatro marcos de página disponibles para el programa y que están inicialmente vacíos.

REF.	0x3A0	0x1B8	0x200	0x320	0x40C	0x6F8	0x504	0x322	0x12C	0x43A	0x380	0x6D2	0x502
#pag													
m0													
m1													
m2													
m3													

Cuestión 7. (0,75 puntos) Considere el siguiente código

```c
int run_now = 1;
char message[] = "Hello World";

void *thread_function(void *arg) {
  int print_count2 = 0;

  while(print_count2 < 4) {
      if (run_now == 2) {
        printf("2");
        run_now = 1;
        print_count2++;
      }
      else {
        sleep(1);
      }
  }

  sleep(3);
}
```

```c
int main() {
  int res;
  pthread_t a_thread;
  void *thread_result;
  int print_count1 = 0;
  if (fork() == 0) {
       res = pthread_create(&a_thread,NULL,
             thread_function, (void *) message);
       while(print_count1 < 4) {
           if (run_now == 1) {
               printf("1");
               run_now = 2;
               print_count1++;
           }
           else {
               sleep(1);
           }
       }
  }
  else {
       execlp("/bin/echo", "/bin/echo", message, NULL);
  }
  printf("\nWaiting for thread to finish...\n");
  res = pthread_join(a_thread, &thread_result);
  printf("Thread joined\n");
  exit(EXIT_SUCCESS);
}
```

Asumiendo que el proceso original tiene PID 100 (y es hijo de *Init*) y que al hijo se le asigna el PID 101, explique razonadamente cuántos hilos se llegan a ejecutar concurrentemente y qué mensajes se mostrarán por pantalla. Indique si existen varias alternativas.

Cuestión 8. (0,75 puntos) Un proceso en UNIX ejecuta el siguiente código:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(void)
{
        int fd1,fd2,i,pos;
        char c;
        char msg[4];

        fd1 = open("outfile.txt", O_CREAT | O_TRUNC | O_RDWR, S_IRUSR | S_IWUSR);
        for (i=0; i < 3; i++) {
                write(fd1, "Init", 4);
                pos = lseek(fd1, 0, SEEK_CUR);
                if (fork() == 0) {
                        /* Hijo */
                        fd2 = open("outfile.txt", O_WRONLY);
                        lseek(fd2, pos, SEEK_SET);
                        sprintf(msg, "Iam%d", i);
                        write(fd2, msg, 4);
                        close(fd2);
                        exit(0);
                } else {
                        /* Padre */
                        wait(NULL);
                }
        }
        lseek(fd1, 0, SEEK_SET);
        printf("File contents are:\n");
        while (read(fd1, &c, 1) > 0)
                printf("%c", (char) c);
        printf("\n");
        close(fd1);
        exit(0);
}
```

Explique qué es lo que hace el programa y si depende del orden en que se ejecuten los procesos. Indique lo que se mostrará por pantalla. NOTA: Se recuerda que la función lseek() devuelve la nueva posición del puntero de posición (bytes desde el comienzo del fichero).

SISTEMAS OPERATIVOS - PROBLEMAS
12 de Febrero de 2016

Nombre _____ DNI _____

Apellidos _____ Grupo _____

Problema 1. (2 puntos) Un sistema de ficheros de un SO diseñado a partir de UNIX utiliza bloques de disco de 16 bytes de capacidad. Para el direccionamiento de estos bloques se utilizan punteros de 8 bits. Cada nodo-i tiene 2 punteros de direccionamiento directo y 1 puntero indirecto simple. Parte del contenido del sistema de ficheros está indicado a continuación:

Mapa de bits (de bloques de datos. '1' es ocupado. El primer índice se corresponde con el bloque 0): 0001 0111 100 1010 0000…0

Tabla de nodos-i (el nodo-i 2 es la raíz del árbol)

nodo-i	2	3	4	5	6	7	8	9	10
Enlaces	NA	NA	NA						
Tipo F/D	D	D	D						
Directo 1	3	5	6						
Directo 2	nil	nil	nil						
Indirecto	nil	nil	nil						

Lista de bloques relevantes:

Bloque 3	
.	2
..	2
opt	3
tmp	4

Bloque 5	
.	3
..	2

Bloque 6	
.	4
..	2

A partir de ese sistema inicial, se ejecutan los siguientes comandos[1]:

```
> cd /
> mkdir tmp/var
> echo "And the rest is rust and stardust."> /opt/LoL
> ln /opt/LoL /tmp/itanab
> ln -s /opt/LoL /tmp/var/okov
```

a. (1 puntos) Indique un posible estado final del sistema de ficheros **explicitando el contenido** del mapa de bits, los nodos-i y de todos los bloques de datos empleados tras ejecutar los comandos. Dibuje también el árbol de directorios

b. (0.5 puntos) A partir del estado anterior, indique qué partes del disco se modificarían al ejecutar la orden

```
> mv /opt/LoL /tmp/var/Humbold
```

c. (0.5 puntos) Considere el siguiente código e indique, para cada llamada al sistema marcada en negrita, el resultado que producen (indicando si habrá error) y cómo afecta al sistema de ficheros anterior (qué partes se modifican, cómo…):

```
….
char buf[25] = {0};
int fd = open("/tmp/itanab",O_RDWR);
lseek(fd,30,SEEK_SET);
read(fd,buf,25);
write(fd,buf,4);
close(fd);
```

[1] La orden *echo "And the rest is rust and stardust."* escribirá 35 bytes: 34 caracteres más retorno de carro.

Problema 2. (2 puntos) Se desea implementar un sistema concurrente para gestionar el servicio de una sucursal bancaria que consta de N cajeros automáticos. Cada cajero está provisto de un cartel luminoso para indicar el turno del cliente al que se asigna (o nada si no hay ningún cliente asignado). Un número arbitrario de clientes (modelados como hilos del programa concurrente) se comportan del siguiente modo:

```
void Cliente(){
    int cajero=cogerTurnoYEsperar()
    ... hacer gestiones oportunas en el cajero ...
    liberarCajero(cajero);
}
```

Cuando un cliente entra en la sucursal debe coger turno (número entero) y a continuación esperará hasta que se muestre su turno en el cartel luminoso de alguno de los cajeros. Para ello, el cliente invocará la función `cogerTurnoYEsperar()` que bloqueará al cliente (si es necesario) y retornará el número de cajero asignado. Cuando esto ocurra, el cliente se acercará al cajero asignado para hacer las gestiones necesarias. Al finalizar con sus gestiones, el cliente invocará la función `liberarCajero()` para notificar al sistema de que dicho cajero queda libre para otro cliente. Para ofrecer un servicio justo a los clientes, el sistema debe garantizar lo siguiente:

- Los cajeros automáticos se deben asignar a cada cliente en orden de llegada (según el turno adquirido)
- Dos clientes no pueden usar el mismo cajero simultáneamente
- Si hay cajeros libres, un cliente en espera debe poder usar el cajero inmediatamente (el sistema debe asignar al cliente un cajero libre lo antes posible)

Implemente las funciones `cogerTurnoYEsperar()` y `liberarCajero()` utilizando mutexes, variables condicionales y otras variables compartidas. Para manipular el estado de los carteles luminosos de cada cajero desde ambas funciones, suponga que se proporciona la función `muestraNumeroCartel(cajero,num)`. Dicha función acepta dos parámetros: el número de cajero, y el número a mostrar (positivo). Si dicho valor es negativo el luminoso se apagará. Considere que los carteles están apagados inicialmente.