	Sistemas Operativos Ingeniería Técnica en Informática de Gestión Curso 2008/2009 EXAMEN FINAL – 13/02/2008	PROBLEMA 1 Valoración: 1 punto

APELLIDOS:_____ **NOMBRE:**_____
D.N.I.:_____

Considere un disco con 5000 cilindros, numerados de 0 hasta 4999. La cabeza de lectura está atendiendo una solicitud al cilindro 143, y la anterior solicitud fue al cilindro 125. La cola de las solicitudes pendientes es:

86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130

Comenzando desde la posición actual de la cabeza, cual es la distancia total (en cilindros) que el brazo del disco deberá recorrer para atender todas las solicitudes pendientes, por cada uno de los siguientes algoritmos de scheduling de disco.

Apartado A. FCFC (0.25 puntos)

Scheduling: 143 ... 86 ...1470 ... 913 ... 1774 ... 948 ... 1509 ... 1022 ... 1750 ... 130
Distancia: 57+ 1384 + 557+ 861 + 826 + 561 + 487 + 728 + 1620 = 7081

Apartado B. SSTF (0.25 puntos)

Solicitudes ordenadas = 86... 130...913...948...1022...1470...1509...1750...1774

Scheduling: 143 ... 130 ... 86 ... 913 ... 1022 ... 1470 ... 1509 ... 1750 ... 1774
Distancia: 13 + 44 + 827 + 109 + 448 + 49 + 241 + 24 = 1755

Apartado C. C-SCAN (0.25 puntos)

Solicitudes ordenadas = 86... 130...913...948...1022...1470...1509...1750...1774

Scheduling: 143 ... 913 ... 948 ... 1022 ... 1470 ... 1509 ... 1750 ... 1774 ... 4999 ... 0 ... 86 ... 130

Distancia: 770 + 35 + 74 + 448 + 39 + 241 + 24 + 3225 + 4999+86+44 = 9985

Apartado D. LOOK (0.25 puntos)

Solicitudes ordenadas = 86... 130...913...948...1022...1470...1509...1750...1774

Scheduling: 143 ... 913 ... 948 ... 1022 ... 1470 ... 1509 ... 1750 ... 1774 ... 130 ... 86

Distancia : 770 + 35 + 74 + 448 + 39 + 241 + 24 + 1644 + 44 = 3319

	Sistemas Operativos **Ingeniería Técnica en Informática de Gestión** **Curso 2008/2009** **EXAMEN FINAL – 13/02/2008**	**PROBLEMA 2** **Valoración: 2 puntos**

APELLIDOS:_____ **NOMBRE:**_____
D.N.I.:_____

Considere una administración de la memoria basada en intercambio (swapping) que utiliza un mapa de bits para representar la ocupación de la memoria. Suponiendo que el mapa de bits sea este:

0111110001100

Suponiendo ahora que al administrador de memoria lleguen las siguientes peticiones en secuencia:

1. Eliminación del proceso que ocupa las unidades 1-3
2. Traer en memoria un nuevo proceso de tamaño 1 (requiere una unidad de asignación).
3. Traer en memoria un nuevo proceso de tamaño 4.
4. Traer en memoria un nuevo proceso de tamaño 4.

Suponiendo que el administrador de memoria utilice el algoritmo de asignación *best fit* (mejor ajuste), ¿cómo se comporta el administrador de memoria y como varía el contenido del mapa de bits?

1. Eliminacion

0001110001100

2. Traer proceso de tamaño 1

0001110001110

Las solicitudes para procesos de tamaño 4 no se pueden satisfacer debido a la falta de bloques libres lo suficientemente grandes.

	Sistemas Operativos **Ingeniería Técnica en Informática de Gestión** **Curso 2008/2009** **EXAMEN FINAL – 13/02/2008**	**PROBLEMA 3** **Valoración: 2 puntos**

APELLIDOS:_____ **NOMBRE:**_____
D.N.I.:_____

Considérese un programa constituido por dos hilos de ejecución que comparten tres semáforos y una variable entera, como se muestra a continuación:

Var semaphore S=M=1 T=0
int x=0

Thread A	**Thread B**

{
while (true) do
begin
down(S**);**
down(M**);**
x**:=**x+1;
up(M**);**
up(T**);**
end
}
	{
	while (true) do
	begin
	down(T**);**
	down(M**);**
	write(x**);**
	x**:=**x+1;
	up(M**);**
	up(S**);**
	end
	}

El programa es ejecutado en un ordenador con una sola CPU.

Apartado A (0.2 puntos)
¿Cuáles son las regiones críticas del programa?

Cuando se modifica X en Thread A

```
// Entra a la region crítica
down(S);
down(M);
x:x+1   // se modifica x
up(M); // libera la variable
up(T)
```

Cuando se modifica X en Thread B

```
// Entra en la region crítica
down(T);
down(M);
x:x+1   // se modifica x
up(M); // libera la variable
up(S)
```

Apartado B (0.2 puntos)
¿Se logra la exclusión mutua? ¿Por qué?

Thread A

Porque el semaforo M bloque X para ser modifica

```
down(M);
x:x+1   // se modifica x
up(M); // libera la variable
```

Thread B

Porque el semaforo M bloque X para ser modifica

```
down(M);
x:x+1   // se modifica x
up(M); // libera la variable
```

Apartado C (0.8 puntos)
¿Cuál es la salida del programa? ¿Por qué?

En el primer ciclo la salida seria un 2

porque Thread A suma 1 a x que inicialmente vale 0 y luego Thread B suma 1 al resultado de haber pasado por Thread A

y en los siguientes ciclos se sumarian otros 2 sucesivamente

Apartado D (0.8 puntos)
Con respecto a los anteriores apartados, ¿qué cambia si los semáforos son inicializados de esta forma S=M=T=1?

Si todos los semáforos se inicializan a 1, entonces:
Tanto el Hilo A como el Hilo B podrían intentar entrar en sus regiones críticas al mismo tiempo, lo que podría llevar a condiciones de carrera.

Esto se debe a que el semáforo T ya no bloquearía inicialmente al Hilo B hasta que el Hilo A haya incrementado x al menos una vez.

Además, el semáforo S ya no bloquearía inicialmente al Hilo A, permitiendo que ambos hilos intenten acceder a la región crítica simultáneamente. Esto podría resultar en una salida impredecible del programa.

	Sistemas Operativos **Ingeniería Técnica en Informática de Gestión** **Curso 2008/2009** **EXAMEN FINAL – 13/02/2008**	**PROBLEMA 4** **Valoración: 0.5 puntos**

APELLIDOS:_____ NOMBRE:_____
D.N.I.:_____

Describir el mecanismo de acceso a páginas en un sistema con memoria virtual que utiliza un TLB.

Cuando la CPU genera una dirección virtual, esta se divide en dos partes:

- Número de página
- Desplazamiento dentro de la página.

El número de página se utiliza como índice en la tabla de páginas para encontrar la entrada correspondiente.

Antes de buscar en la tabla de páginas, el sistema operativo primero verifica si la entrada de la página está en el TLB.

- Si la entrada está en el TLB (un “acceso TLB”), entonces se utiliza la entrada del TLB para traducir la dirección virtual a una dirección física.
- Si la entrada no está en el TLB (un “fallo TLB”), entonces el sistema operativo debe buscar en la tabla de páginas en la memoria principal.
  - Una vez que se encuentra la entrada de la página en la tabla de páginas, se carga en el TLB para futuros accesos.
  - Finalmente, el desplazamiento se añade a la dirección base del marco de página para obtener la dirección física completa.

Este mecanismo permite un acceso rápido a las entradas de la tabla de páginas que se utilizan con frecuencia, mejorando así el rendimiento general del sistema

	Sistemas Operativos **Ingeniería Técnica en Informática de Gestión** **Curso 2008/2009** **EXAMEN FINAL – 13/02/2008**	**PROBLEMA 5** **Valoración: 1.5 puntos**

APELLIDOS:_____ NOMBRE:_____
D.N.I.:_____

Completar (bien en pseudo código o bien en C), el siguiente código para poder resolver el problema de productor consumidor, suponiendo tener los métodos queueAdd para agregar un elemento al buffer, y queueDel para borrar un elemento del buffer. Además, la funciones queueInit y queueDelete son utilizadas para inicializar y borrar el buffer.

```
#include <pthread.h>
#include <semaphore.h>
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define QUEUESIZE 4      //el buffer tiene tamaño 4
#define LOOP 20         //cuantas veces hay que ejecutar el loop en el productor
y en el consumidor

void *producer (void *args);
void *consumer (void *args);
//SEMÁFORO BINARIO
pthread_mutex_t mut =  PTHREAD_MUTEX_INITIALIZER;
//SEMÁFOROS GENÉRICOS
sem_t notFull;
sem_t notEmpty;

//estructura del buffer
typedef struct {
      int buf[QUEUESIZE];
      long head, tail;
      int full, empty;

} queue;

queue *queueInit (void);                    //inicializa el buffer
void queueDelete (queue *q);   //borra el buffer
void queueAdd (queue *q, int in);   //agrega un elemento al buffer
void queueDel (queue *q, int *out);       //saca un elemento del buffer

int main ()
{
      //inicialización semáforos
      sem_init(&notEmpty, 0, 0);
      sem_init(&notFull, 0 , QUEUESIZE);
      queue *fifo;
      pthread_t pro, con;
      fifo = queueInit ();
      if (fifo ==  NULL) {
          fprintf (stderr, "main: Queue Init failed.\n");
          exit (1);
      }
      //COMPLETAR
```

```
        pthread_create (&pro, NULL, producer, fifo);
        pthread_create (&con, NULL, consumer, fifo);

        pthread_join (pro, NULL);
        pthread_join (con, NULL);
```

```
        //borra el buffer
        queueDelete (fifo);
        return 0;
}

void *producer (void *q)
{
        queue *fifo;
        int i;
        fifo = (queue *)q;
        //COMPLETAR
```

```
        for (i = 0; i < LOOP; i++) {
            pthread_mutex_lock (&mut);
            while (fifo->full) {
                printf ("producer: queue FULL.\n");
                pthread_mutex_unlock (&mut);
                sleep (1);
            }
            queueAdd (fifo, i);
            printf ("producer: added %d\n", i);
            pthread_mutex_unlock (&mut);
            sem_post (&notEmpty);
        }
```

```
}

void *consumer (void *q)
{
    queue *fifo;
    int i, d;
```

```
    fifo = (queue *)q;
    //COMPLETAR
```

```
        while(1) {
            sem_wait (&notEmpty);
            pthread_mutex_lock (&mut);
            while (fifo->empty) {
                printf ("consumer: queue EMPTY.\n");
                pthread_mutex_unlock (&mut);
                sleep (1);
            }
            queueDel (fifo, &d);
            printf ("consumer: deleted %d\n", d);
            pthread_mutex_unlock (&mut);
            sem_post (&notFull);
        }
```

```
}
```

	Sistemas Operativos **Ingeniería Técnica en Informática de Gestión** **Curso 2008/2009** **EXAMEN FINAL – 13/02/2008**	**PROBLEMA 6** **Valoración: 0.5 puntos**

APELLIDOS:_____ NOMBRE:_____
D.N.I.:_____

¿Cuáles son las posibles formas de efectuar E/S? Describir brevemente cada una.

E/S Programada (PIO - Programmed Input/Output):

Este método requiere que el procesador ejecute comandos para cada transferencia de E/S, por lo que es el método más lento. Sin embargo, las E/S programadas son las más sencillas de programar.

E/S Controlada por Interrupción:

Este método permite a los dispositivos de E/S notificar al sistema operativo cuando están listos para realizar una operación o cuando se produce algún evento importante.

Acceso Directo a Memoria (DMA - Direct Memory Access):

Este método permite a los dispositivos de E/S acceder directamente a la memoria del sistema sin la intervención del procesador.

E/S Sincrónica:

En este método, la E/S se realiza de manera que el proceso que realiza la operación de E/S se bloquea hasta que la operación se completa.

E/S Asincrónica:

En este método, la E/S se realiza de manera que el proceso que realiza la operación de E/S puede continuar ejecutándose mientras la operación de E/S se está realizando