
 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Examen de la convocatoria extraordinaria 21 de junio de 2011</p>	
---	--	---

ATENCIÓN:

- Lea atentamente todo el enunciado antes de comenzar a contestar.
- Dispone de 2 horas y 1/2 para realizar la prueba.
- No se podrán utilizar libros ni apuntes, ni calculadoras de ningún tipo.
- Los teléfonos móviles deberán permanecer desconectados durante la prueba (apagados, no silenciados).
- Solamente se corregirán los ejercicios contestados con bolígrafo. Por favor no utilice lápiz.

APELLIDOS:

NOMBRE:

NIA:



GRUPO:

Ejercicio 1. Teoría [2,5 puntos]:

Pregunta 1. Haga un esquema y explique la estructura de un sistema operativo por capas, como Windows.

Pregunta 2. Explique cómo se trata en Linux una señal como SIGINT.

Pregunta 3. ¿Cuál es la diferencia entre un semáforo de Linux y un mutex?

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Examen de la convocatoria extraordinaria 21 de junio de 2011</p>	
---	--	---

Ejercicio 2 [2,5 puntos]:

Se desea realizar una aplicación en C que conste de un proceso padre y varios hijos.

El proceso padre debe, cada vez que reciba la señal SIGUSR1 crear un proceso hijo y mostrar cuantos hijos están vivos en ese momento. El padre estará en un bucle infinito esperando que finalicen los hijos o que se reciba la señal .

Los hijos deben estar vivos durante un tiempo aleatorio entre 5 y 20 segundos y después deben finalizar.

Se pide:

1. Realizar el código del proceso hijo.
2. Programar el código del proceso padre que incluirá:
 - El tratamiento de la señal de CTRL-C y la creación de los hijos
 - El programa principal con un bucle que espere por los hijos que han terminado.

Ejercicio 3 [2,5 puntos]:

El siguiente programa ofrece una solución basada en semáforos para el problema de productor-consumidor

```
int BufferSize = . . . ;



semaphore mutex = . . . ;
semaphore empty = . . . ;
semaphore full = . . . ;

producer()
{
    int item;

    while (TRUE) {
        make_new(item);
        down(&empty);
        down(&mutex);
        put_item(item);
        up(&mutex);
        up(&full);
    }
}

consumer()
{
    int item;

    while (TRUE) {
        down(&full);
        down(&mutex);
        remove_item(item);
        up(&mutex);
        up(&empty);
        consume_item(item);
    }
}
```

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Examen de la convocatoria extraordinaria 21 de junio de 2011</p>	
---	--	---

Se pide:



- Inicialice la variable BufferSize y los 3 semáforos, asumiendo que el buffer tiene una capacidad máxima de almacenamiento de 10 objetos.
- ¿Es el semáforo mutex necesario? Justifique su respuesta.
- ¿Es posible simplificar el programa utilizando solamente un semáforo count en vez de los semáforos full and empty? Justifique su respuesta. Si es posible, describa la solución.
- En la solución proporcionada, hay algún problema si se intercambian las llamadas `down(&empty)` y `down(&mutex)` en el productor?

Ejercicio 4 [2,5 puntos]:

Se tiene un disco con sistema de ficheros Unix. El tamaño del bloque de ficheros es de 1KByte, las direcciones a los bloques son de 4 bytes y los i-nodos tienen la estructura tradicional (10 apuntadores directos, 1 indirecto simple, 1 indirecto doble y 1 indirecto triple).

Se pide:

- Escribir un programa que lea 8200 KBytes con acceso a nivel de 1 KByte.
- Calcular cuántos bloques de disco incluyendo el i-nodo y los bloques de direcciones y datos se estarán utilizando para almacenar un fichero de 8200 KBytes. Explique detalladamente cómo se han realizado los calculos

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Examen de la convocatoria extraordinaria 21 de junio de 2011</p>	
---	--	---

SOLUCIONES:

Ejercicio 1.

Pregunta 1. Haga un esquema y explique la estructura de un sistema operativo por capas, como Windows.

Pregunta 2. Explique cómo se trata en Linux una señal como SIGINT.

Pregunta 3. ¿Cuál es la diferencia entre un semáforo de Linux y un mutex?

Ejercicio 2

```
#include <signal.h>
#include <stdio.h>
#include <unistd.h>



int crearhijo=0;

void capturaSenal (int s){
    crearhijo=1;
}

int hijo (){
    int espera;
    srand (getpid());
    espera=random ()%16 + 5;
    printf ("Hijo %d espera %d segundos\n", getpid(), espera);
    sleep (espera);
    _exit (0);
}

main (){
    struct sigaction sa1;
    int pidhijo;
    int conthijos=0;

    sa1.sa_handler=capturaSenal;
    sa1.sa_flags=0;
    sigemptyset(&(sa1.sa_mask));
    sigaction (SIGUSR1, &sa1,NULL);
    // while (crearhijo==0)
    // pause();
    while (1){
        while (crearhijo==0)
```

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Examen de la convocatoria extraordinaria 21 de junio de 2011</p>	
---	--	---

```

    if (pidhijo=wait (NULL) >0){
        conthijos--;
    }
    else {
        sleep(1);
    }

    if (fork() == 0)
        hijo ();
    conthijos++;
    crearhijo=0;
    printf ("Hay %d hijos vivos\n",conthijos);
}
}

```

Ejercicio 3

```

a)
int BufferSize = 10;

    semaphore mutex = 1;           // Controls access to critical
section
    semaphore empty = BufferSize;  // counts number of empty buffer
slots
    semaphore full = 0;           // counts number of full buffer
slots

```



b) El mutex es necesario para implementar la sección crítica. Si no se utilizara el mutex, se podría producir una condición de carrera, que resultaría en un comportamiento no deseado del programa.

c) No es posible implementar la solución con un semáforo, porque hay 2 situaciones en el que el programa se tiene que bloquear (buffer lleno y buffer vacío), mientras que un semáforo se puede bloquear solamente cuando se llama wait and su valor es 0.

d) Hay un problema, podría producirse un interbloqueo.

Por ejemplo:

1. Cuando el buffer está vacío, empty=0.
2. El productor llama wait(&mutex) => mutex=0.
3. El productor llama wait(&empty) y se bloquea.
4. El consumidor llama wait(&mutex) y se bloquea.
5. Los 2 están bloqueados=> interbloqueo.

 <p>Universidad Carlos III de Madrid</p>	<p>Departamento de Informática Grado en Ingeniería Informática Sistemas Operativos</p> <p>Examen de la convocatoria extraordinaria 21 de junio de 2011</p>	
---	--	---

Ejercicio 4

Solución apartado 1.

?????

Solución apartado 2.

- Necesitamos 1 bloque para almacenar el i-nodo.
- Con los 10 apuntadores directos estaremos almacenando 10 KBytes = 10 bloques
- En cada bloque de direcciones caben 256 direcciones de bloque: 1KBytes / 4 bytes = 256 direcciones.
- Con el puntero indirecto simple apuntaremos a 1 bloque que contiene 256 direcciones que apuntan a 256 bloques que almacenan 256 KBytes
- Con el puntero indirecto doble apuntaremos a 1 bloque que contiene 256 direcciones que pueden apuntar a 256 bloques de direcciones que apuntan cada uno de ellos a 256 bloques de datos para ver cuantos bloques nos faltan realizamos el siguiente cálculo:
8200 Kbytes 10 Kbytes de los punteros simples 256 Kbytes del puntero indirecto simple = 7934 KBytes que nos faltan por almacenar

$$7934 \text{ Kbytes} / (256 \text{ bloque} * 1 \text{ Kbytes por bloque}) = 3099$$

luego necesitamos tener 31 direcciones que apunten cada una de ellas a 31 bloques que contienen 256 direcciones que apuntan a 256 bloques de datos cada una que nos da un total de 7936 Kbytes, luego el último bloque de direcciones tiene las dos últimas sin rellenar, y apuntaríamos a 7934 bloques.

- N° de bloques = 1 i-nodo + 10 bloques directos + 1 bloque indirecto simple + 256 bloques apuntados por el puntero indirecto + 1 bloque indirecto doble + 31 bloques apuntados por el puntero indirectos doble + 7934 bloques apuntados por los punteros indirectos dobles = 8234 bloques