



**SISTEMAS OPERATIVOS. CUESTIONES**  
**Septiembre de 2013.**

Nombre \_\_\_\_\_ DNI \_\_\_\_\_

Apellidos \_\_\_\_\_ Grupo \_\_\_\_\_

**Cuestión 1. (0,75 puntos)** Un sistema de ficheros UNIX utiliza bloques de 1024 bytes y direcciones de disco de 32 bits. Los nodos-i contienen 8 direcciones de disco para bloques de datos y una dirección de bloque índice indirecto simple. ¿Cuál es el tamaño máximo de un fichero en este sistema? Un programa UNIX crea un fichero en este sistema e inmediatamente después escribe un byte de datos en la posición 1.000 y otro en la posición 10.000. ¿Cuántos bloques de datos ocupa este nuevo fichero en disco?

**Cuestión 2. (0,75 puntos)** Un disco tiene las siguientes peticiones pendientes: 45, 132, 20, 23, 198, 170, 180, 78. La cabeza de lectura/escritura está inicialmente sobre el cilindro 53 y su movimiento es hacia cilindros crecientes. Indique el orden de atención de dichas solicitudes ante los siguientes algoritmos:

	1	2	3	4	5	6	7	8
FCFS								
SCAN								
C-SCAN								

**Cuestión 3. (0,75 puntos)** Un computador dispone de un reloj programable que genera marcas a 1 Hz. Para controlar la hora del día, el computador dispone de un contador de marcas de 32 bits. Si la fecha de referencia de este computador es el 01/01/2013, ¿hasta qué año podremos controlar la hora del sistema?

**Cuestión 4. (0,75 puntos)** En un sistema con memoria virtual paginada dos procesos quieren compartir una página que contiene datos. ¿Deben tener las mismas direcciones virtuales los datos compartidos? ¿Deben compartir entradas de la TP? Dibujar una posible TP para cada proceso si la región de memoria compartida está en el marco de página 33.

**Cuestión 5. (0,75 puntos)** Indicar cómo funciona el algoritmo LRU (información que almacena para cada marco de página, número de fallos de página,...) suponiendo una memoria física de cuatro marcos de página y usando como ejemplo la siguiente cadena de referencias: *a b g a d a d e g b d e*.

**Cuestión 6. (0,75 puntos)** Implemente las siguientes operaciones de una variable condicional usando semáforos generales y cerrojos:

```
cond_wait(mi_cond_t *c,mutex_t mut);  
/* El hilo que llama a esta función  
es el propietario del mutex (no es  
necesario comprobarlo en este  
código)*/
```

```
cond_broadcast (mi_cond_t *c);  
/* Despierta a todos los hilos que  
estén esperando en la var. condicional  
*/
```

Especifique el tipo de datos *mi\_cond\_t* y proponga la implementación de *cond\_wait* y *cond\_broadcast*.

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

```
char buf1[4] = "AAAA";
char buf2[4] = "BBBB";
char buf3[4] = "CCCC";
int fd1;

int main() {
    if ( fork() == 0 ) {
        fd1=open( "prueba",O_RDWR | O_CREAT, 0666 );
        write(fd1,buf1,4);
        write(fd1,buf2,4);
        close(fd1);
    }
    else {
        fd1=open( "prueba",O_RDWR | O_CREAT, 0666 );
        wait(NULL);
        write(fd1,buf3,4);
        lseek(fd1,0,SEEK_SET);
        read(fd1,buf2,4);
        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? Y, ¿cuál será el contenido de la variable *buf2*? **Justifique la respuesta.**

**Cuestión 8. (0,75 puntos)** Estudie el siguiente código e indique qué se mostrará por pantalla tras su ejecución:

```
int a=0;
main() {
    int b=1;
    for (int i=0; i < 3; i++) {
        if (fork() == 0) {
            b=b+2;
            thread_create(&tid,suma,b);
            thread_join(tid);
            exit(0);
        }
        else {
            wait(NULL);
            a=a+b;
            b=b+1;
        }
        printf("i=%d a=%d, b=%d\n",i,a,b);
    }
}

void suma (int num) {
    a = num + a;
}
```

N.B. Se ha simplificado el paso de parámetros a un hilo para facilitar la comprensión del código. La interfaz POSIX especifica que ese argumento debe ser un puntero. Para este ejercicio, considera que el valor actual de la variable *b* es lo que se pasa como argumento al hilo recién creado

	<b>SISTEMAS OPERATIVOS. PROBLEMAS</b> <b>Septiembre 2013.</b>
	Nombre _____ DNI _____ Apellidos _____ Grupo _____

**Problema 1. (2 puntos)** Un sistema de ficheros basado en i-nodos y mapa de bits contiene la siguiente información:

Mapa de bits: 00010110001000010110...0

Nodos-i:

Info/Nodo-i	N2	N3	N4	N5	N6
Tamaño	1		1	1	2
#Enlaces	NA	NA	NA	NA	
Tipo F/D	D	D	D	D	F
Directo	3	5		10	15
Indirecto	-	-	-	-	17

Bloques:

B3		B5		B6		B10		B15	B17	B18
.	2	.	3	.	4	.	5			Datos
..		..	2	..	2	..	2			
A	3	D	6	E	6	F	6			
B	4									
C	5									

Se pide:

- Rellene los huecos para que el sistema sea consistente. Asuma para ello que el tamaño se expresa en bloques.
- Dibuje el árbol del directorio empleando óvalos para los directorios, rectángulos para los ficheros y triángulos para los datos.

**Problema 2. (2 puntos)** Resolver el siguiente problema de concurrencia denominado “CO<sub>2</sub>”. El programa principal genera constantemente hilos de dos clases: oxígeno y carbono, con el objetivo de formar moléculas de “CO<sub>2</sub>”. A continuación se muestra un esquema del código de los distintos hilos. Por ejemplo, un hilo de carbono deberá esperar a que haya al menos dos oxígeno en el sistema. Tras garantizar que hay dos oxígenos (y el propio carbono) , esos tres hilos se ejecutarán la función “FormarMolecula()”.

<b>main( )</b> {	<b>Carbono( )</b> {	<b>Oxígeno( )</b> {
while(1) {	Si no hay dos oxígenos	Si no hay un O y un C
n=random();	Esperar...	Esperar...
si n es par	En caso contrario	En caso contrario
crear hilo Carbono	Despertar 2 oxígenos	Despertar hilos...
si no		
crear hilo Oxígeno	FormarMolecula()	FormarMolecula()
}	}	}

Codificar las funciones **Carbono** y **Oxígeno** con **mutexes y variables condicionales**.

N.B No es necesario garantizar que únicamente dos oxígenos y un carbono ejecutan concurrentemente la función FormarMolecula(). Sólo hay que resolver el ejercicio de señalización cruzada para que esperen si no hay recursos suficientes.