UNIVERSITAT POLITÈCNICA DE VALÈNCIA
Departamento de Informática de Sistemas y Computadoras (DISCA)

fSO

Ejercicio de Evaluación
05 de Noviembre de 2012

etsinf

APELLIDOS		**NOMBRE**		Grupo
DNI		**Firma**		

- **No desgrape las hojas.**
- **Conteste exclusivamente en el espacio reservado para ello.**
- **Utilice letra clara y legible. Responda de forma breve y precisa.**
- **El examen consta de 9 cuestiones, en cada una de ellas se indica su valoración.**

---

1) ¿Se puede crear una aplicación de usuario cuyo código no incluya ninguna llamada al sistema? Justifique su respuesta. **0,75 puntos**

1	

---

2) Durante la inicialización o arranque del sistema operativo de un computador de sobremesa moderno, el procesador ¿podría encontrarse en modo usuario? Justifique la respuesta. **0,75 puntos**

2	

---

3) El siguiente código corresponde al archivo ejecutable generado con el nombre "Ejemplo1".

```
   /*** Ejemplo1***/
   #include "todas_las_cabeceras_necesarias.h"
1
2  main()
3  { int i=0;
4    pid_t pid, pid2;
5
6
7    while (i<2)
8    { pid=fork()
9       switch(pid)
10       {case (-1):{printf("Error creando hijo\n");
11                   break;}
12        case (0):{pid2=fork();
13                  printf("Hijo %i creado\n",i);
14                  sleep(10);
15                  exit(0);
16                 }
17        default: {printf("Padre\n");
18                  sleep(5);}
19      }
20     i++;
21   }
22   exit(0);
23 }
```

Suponga que "Ejemplo1" se ejecuta correctamente:

a) Indique de forma razonada, el número de procesos que creará y dibuje el árbol de procesos generado.
b) Indique de forma justificada, si pueden producirse procesos zombies y/o huérfanos.

1.5 puntos

3	**a)**
	b)

---

4) Indique de forma justificada si son ciertas o falsas las siguientes afirmaciones sobre los estados de los procesos:

a) Sólo los procesos que están en *Ejecución* pueden pasar al estado *Terminado*, una vez finaliza la ejecución de sus instrucciones.
b) El estado *Suspendido* sólo puede ser alcanzado por aquellos procesos que hayan solicitado o estén realizando una operación de E/S bloqueante
c) El cambio de estado de en Ejecución al de *Preparado* sólo es posible en sistemas con planificadores expulsivos.

1.0 puntos

4	a)
	b)

c)

---

5) Un sistema dispone de un planificador multicolas a corto plazo (PCP) con tres colas Cola0, Cola1 y Cola2, cuyos algoritmos de planificación son **RR con q=1, SRTF, y FCFS**, respectivamente.
La planificación entre **colas es gestionada con prioridades expulsivas** siendo la más prioritaria la Cola2 y la menos prioritaria la Cola0. Cada proceso dispone de un contador de promoción (ContPro) que el sistema mantiene para establecer su promoción entre colas, de manera que los procesos puedan alcanzar colas más prioritarias. Cada vez que un proceso **pasa al estado suspendido** su ContPro se incrementa en 1 (ContPro= ContPro+1). Un proceso es ubicado en la Cola0 si su ContPro es igual a 0, en la Cola1 si su ContPro=1 y en la Cola2 si ContPro>=2. Los procesos que llegan al sistema se les asigna un ContPro=0 y van a la Cola0.
Suponga que las operaciones de E/S se efectúan sobre el mismo dispositivo gestionado con FCFS y que al sistema llegan los procesos mostrados en la tabla:

Proceso	Perfil de ejecución	Instante de llegada	ContPro
A	2 CPU + 2 E/S + 1 CPU + 4 E/S + 1 CPU	0	0
B	1 CPU + 2 E/S + 4 CPU + 2 E/S + 2 CPU	1	0
C	3 CPU + 1 E/S + 1 CPU	2	0
D	2 CPU + 1 E/S+ 1CPU	3	0

a) Rellene la siguiente tabla indicando en cada instante de tiempo donde se encuentran los procesos.
b) Indique la utilización de CPU para esta carga y los tiempos medios de espera y retorno.

2.0 puntos (1.25+0.75)

5a

T	Cola 0 RR q=1	Cola 1 SRTF	Cola 2 FCFS	CPU	Cola E/S	E/S	Evento
0							Llega A , ContPro(A)=0
1							Llega B, ContPro(B)=0
2							LlegaC, ContPro(C)=0
3							Llega D, ContPro(D)=0
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							

5b	Tiempo medio de espera= Tiempo medio de retorno= Utilización de CPU

---

6) Dado el siguiente código cuyo archivo ejecutable ha sido generado con el nombre "Ejemplo2".

```
 1 /*** Ejemplo2***/
 2 #include <stdio.h>
 3 #include <pthread.h>
 4
 5 void *fun_hilo( void *ptr )
 6 { int sec;
 7
 8   sec=(int)ptr;
 9   sleep(sec);
10   printf("Yo he esperado %d segundos\n",sec);
11 }
12
13 int main()
14 {
15   pthread_attr_t atrib;
16   pthread_t hilo1, hilo2, hilo3;
17
18   pthread_attr_init( &atrib );
19   pthread_create( &hilo1, &atrib, fun_hilo, (void *)30);
20   pthread_create( &hilo2, &atrib, fun_hilo, (void *)1);
21   pthread_create( &hilo3, &atrib, fun_hilo, (void *)10);
22   pthread_join( hilo3, NULL)
23 }
```

Indique las cadenas que imprime el programa en la terminal tras su ejecución. Justifique su respuesta

1.0 puntos

6	

7) El siguiente código C corresponde a un proceso con dos hilos que comparten memoria y se han de sincronizar utilizando el **mecanismo de espera activa**. Para diseñar su **protocolo de entrada y salida** a la sección crítica utilizan las variables compartidas **flag** y **turn** declaradas.

```
1  //***** Ejemplo 3 **//
2  #include <pthread.h>
3  …
4  int flag[2];
5  int turn;
6
7  void* thread(void* id)
8  { int i;
9    i = (int)id;
10
11  while ( 1 ) {
12
13     remaining_section();
14
15     /**b)Asigne valores a flag[i] y a turn**/
16     while(/**c)Condición del Protocolo de Entrada**/);
17
18     critical_section();
19
20     flag[i] = 0;
21   }
22 }
23 int main() {
24   pthread_t th0, th1;
25   pthread_attr_t atrib;
26   pthread_attr_init(&atrib);
27   /**a)Inicialice la variable flag **/
28   pthread_create(&th0, &atrib, thread, (void *)0);
29   pthread_create(&th1, &atrib, thread, (void *)1);
30   pthread_join(th0, NULL);
31   pthread_join(th1, NULL);
32 }
```

Se pide:

a) Reemplace "`/**a)Inicialice la variable flag **/`" por el código C correspondiente.
b) Reemplace "`/**b)Asigne valores a flag[i] y a turn**/`" por el código C correspondiente.
c) Reemplace "`/**c)Condicion del protocolo de entrada**/`" por el código C correspondiente.

1.0 puntos

7	**a)**
	b)
	c)

8) Indique todos los posibles valores que puede alcanzar la variable x tras la ejecución concurrente de los procesos A, B y C. Justifique su respuesta indicando para cada uno de valores el orden de ejecución de las distintas secciones.

// Variables compartidas
int x=1;
Semáforos S1=3, S2=0, S3=0;		

// Proceso A	//Proceso B	//Proceso C

P(S1)
P(S3)
x = 2*x + 1; // sección 1
V(S2)
	P(S1)
	P(S2)
	x = x*3; // sección 2
	V(S1)
		P(S1)
		x = x + 2; // sección 3
		V(S3)
		P(S1)
		x = x + 3; // sección 4

1.0 puntos

8	

9) Complete en "Ejemplo4" el código de las funciones fth_UNO y fth_DOS, con las operaciones sobre **semáforos** necesarias, para que al ejecutarlo muestre por pantalla de manera ordenada, los diez primeros números enteros, es decir, " 0 1 2 3 4 5 6 7 8 9" .

```
 1  /*** Ejemplo4***/
 2  #include <stdio.h>
 3  #include <pthread.h>
 4
 5  void *fth_UNO( void *ptr )
 6  { int i;
 7    for (i=1; i<10; i+=2)
 8    {
 9      printf("%d ",i);
10
11    }
12  }

13  void *fth_DOS( void *ptr )
14  { int i;
15  for (i=0; i<10; i+=2)
16    {
17      printf("%d ",i);
18
19    }
20  }

21  int main()
22  { pthread_attr_t atrib;
23    pthread_t th1, th2;
24
25    pthread_attr_init( &atrib );
26    pthread_create(&th1,&atrib,fth_UNO, NULL);
27    pthread_create(&th2,&atrib,fth_DOS, NULL);
28    pthread_join( th1, NULL);
29    pthread_join( th2, NULL);
30    printf ("\n");
  }
```

El hilo "th1" debe mostrar los números impares y "th2" los pares. Para sincronizar la ejecución de los hilos utilice tantos semáforos como sea necesario y diga su valor de inicialización. Utilice la nomenclatura P y V.

1.0 puntos

9	