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

fSO

etsinf

Ejercicio de Evaluación
07 de Noviembre de 2011

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, la 6 vale 2 (1.2+0.8) puntos y el resto 1 punto cada una.**

Diga en qué consiste la técnica de multiprogramación e indique las ventajas de su uso. Analice si dicha técnica tiene o no sentido en el caso de que los procesos a ejecutar tuviesen una única ráfaga de CPU y ninguna de entrada salida, suponga que no hay interacción usuario máquina

1	

Dado un computador dotado de sistema operativo

a) Justifique la necesidad de que el procesador disponga de al menos dos modos de funcionamientos
b) Indique para cada una de las acciones propuestas cuales se realizan en modo núcleo y cuales en modo usuario

2

a)

b) Acciones Propuestas	**Núcleo**	**Usuario**
Programar el controlador del disco		
Seleccionar un proceso de la cola de preparados		
Invocar una llamada al sistema		
Leer 256 bytes de un fichero		
Ejecutar instrucciones que contiene expresión matemáticas complejas		

Dada las siguientes transiciones entre procesos indique de forma justificada si son o no posibles exponiendo un ejemplo cuando sea posible:

3	**En ejecución a preparado**
	De preparado a suspendido
	De suspendido a en ejecución

Dado el siguiente código cuyo fichero ejecutable ha sido generado con el nombre "Ejemplo1".

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

Indique de forma justificada:

a) El número de procesos que se generan al ejecutarlo y el parentesco existente entre ellos.
b) Indique que mensajes se mostrarán en pantalla, si la llamada fork() siempre tiene éxito.

4	**a)**
	b)

Dado el siguiente código cuyo fichero ejecutable ha sido generado con el nombre "Ejemplo1".

```
   /*** Ejemplo1***/
 1 #include "todas_las_cabeceras_necesarias.h"
 2
 3 int main()
 4 {
 5   int status;
 6   printf("Mensaje 1: antes de  exec()\n");
 7   if (execl("/bin/ps","ps","-la", NULL)<0)
 8      { printf("Mensaje 2: después de  exec()\n");
 9        exit(1);}
10
11   printf("Mensaje 3: antes del exit()\n);
12   exit(0);
   }
```

Indique de forma justificada:

a) Cuantos procesos se pueden llegar a crear durante su ejecución, en que número de líneas se lleva a cabo dicha creación y que mensaje imprime cada uno de ellos.

b) Cuando aparecerán por la salida estándar "*Mensaje 3: antes del exit()*".

5	a)
	b)

Un Sistema Operativo de tiempo compartido acepta trabajos lanzados desde terminales locales (interactivos I), desde una línea de red (R) y trabajos en Batch (B). La gestión de trabajos es por colas multinivel sin realimentación. La política de planificación entre colas es por prioridades expulsivas. La cola con mayor prioridad es la de procesos interactivos (I) y la de menor la de Batch (B). Cada proceso va a la cola de su tipo y permanece en ella durante su vida. La política de cada cola es:

- Interactivos: R-R (q=1)
- Red: R-R (q=2)
- Batch: FCFS

El orden de llegada de los procesos a las colas es el siguiente: en primer lugar los nuevos, a continuación los procedentes de E/S y por último los que salen de CPU. Suponga que todas las operaciones E/S se realizan en un único dispositivo con política de servicio FCFS y que se solicita ejecutar el siguiente grupo de trabajos:

Trabajo	Instante de llegada	Tipo	
T1	0	I	2CPU+2E/S+1CPU+2E/S+2CPU
T2	2	R	4CPU+1E/S+ 4CPU+1 E/S+1 CPU
T3	4	B	1CPU+1E/S+ 5CPU+1 E/S+1CPU
T4	5	I	2CPU+4E/S+1CPU
T5	21	B	2CPU

a) Indique el diagrama de uso de CPU, rellenando la tabla adjunta para cada instante de tiempo.

b) Calcule el tiempo medio de espera en la cola de preparados, el tiempo medio de retorno y la utilización de CPU

6 a

T	Cola I	Cola R	Cola B	CPU	Cola E/S	E/S	Evento
0							Llega T1
1							
2							
3							
4							
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26							
27							
28							

6 b

Dado el problema de la sección crítica y sus posibles soluciones, diga si las siguientes sentencias son verdaderas o falsas:

7	V/F	
		Al conjunto instrucciones de un proceso, que acceden a datos compartidos con al menos otro proceso del sistema y que alguno de dichos procesos puede modificar, se le llama sección critica.
		Una solución correcta al problema de la sección crítica debe garantizar la exclusión mutua de todo el código de los procesos que se ejecutan concurrentemente.
		Una solución propuesta al problema de la sección crítica utiliza la instrucción test_and_set que debe ejecutarse de forma atómica.
		Atómica significa que no puede ejecutarse con espera activa y es necesaria la

		intervención de un mutex
		La condición que debe cumplir cualquier solución válida al problema de la sección critica y que establece que "*si ningún proceso está ejecutando su sección crítica y hay otros que desean entrar a las suyas, entonces la decisión de qué proceso entrará a la sección crítica se ha de tomar en un tiempo finito y sólo depende de los procesos que desean entrar*" se denomina **espera limitada**.
		La soluciones al problema de la sección crítica con espera activa presenta infrautilización del procesador
		Los mutex de POSIX son una solución al problema de la sección crítica sin espera activa

Dado el problema del productor consumidor estudiado en clase donde aparece tanto una sección critica, como una relación de precedencia. Recuerde que los consumidores no deben consumir antes de que los productores hayan producido. Complete el siguiente código realizando operaciones sobre los semáforos *mutex, vacio y lleno*, que ya se encuentran declarados e inicializados. Justifique su solución indicando la utilidad de las operaciones propuestas.

```
#include <semaphore.h>
#define N  20
int  buffer[N];
int  entrada, salida, contador
sem_t mutex, lleno, vacio;
```

```
void *func_prod(void *p) {
  int item;
  while(1) {
    item = producir()

//Complete (1)
……..
    buffer[entrada] = item;
    entrada = (entrada + 1) % N;
    contador = contador + 1;
……
//Complete (2)
  }
}
```

```
void *func_cons(void *p) {
  int item;
  while(1) {

//Complete (3)
    ……
    item = buffer[salida];
    salida = (salida + 1) % N;
    contador = contador - 1;
    ……
//Complete(4)
    consumir(item);
  }
}
```

```
void main ()
{  ……
  sem_init(&mutex,0,1);
  sem_init(&vacio,0,N);
  sem_init(&lleno,0,0);
  ……
}
```

8

a) Solución propuesta

b) Utilidad o función de cada instrucción propuesta

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

```
void * funcion_hilo1(void * arg) {
  sleep(40);
  printf("Soy el hilo 1\n");
  return null;
}
```

```
void * funcion_hilo2(void * arg) {
  sleep(50);
  printf("Soy el hilo 0\n");
  return null;
}
```

```
int main (void) {
  pthread_t th1,th2;
  pthread_attr_t atrib;

  pthread_attr_init( &atrib );

  printf("Erase una vez dos hilos...\n");
  pthread_create( &th1, &atrib, function_hilo1, null);
  pthread_create( &th2, &atrib, function_hilo2, null);
  exit(0);
}
```

9	