Arquitectura y Sistemas Operativos – Plan 2024 – Profesor Agustin Martinez

Segundo parcial – Comisión Nº7 – Fecha 30/05/2025

- ***Condición mínima de aprobación: 60% del examen correctamente resuelto. (6 puntos)***
- ***Todo el contenido debe ser escrito en lapicera para ser corregido.***
- ***Todas las respuestas deben estar justificadas para ser tenidas en cuenta***

Nombre y apellido……DNI……..

1 (2 Puntos)	2 (1 Puntos)	3 (2 Puntos)	4 (3 puntos)	5 (2 puntos)	Calificación

1) Indique si las siguientes sentencias son verdaderas o falsas. JUSTIFICAR LAS FALSAS:

a) Cuando se bloquea un hilo de nivel kernel el núcleo puede planificar otro hilo del mismo proceso. **V**

b) Si el valor del semáforo mutex es -1 quiere decir que hay 1 procesos bloqueados. **F, Mutex solo puede valer 1 o 0.**

c) El Deadlock es una situación en la que un solo proceso espera indefinidamente un recurso. **F, Dos o mas procesos.**

2) Seleccione con una cruz la opción correspondiente según el enunciado:

¿Cuál de los siguientes es uno de los objetivos de MAYOR importancia en el diseño del sistema de E/S?

- Control.
- Comunicación.
- Generalidad. **X**
- Legibilidad.

El algoritmo de Peterson y Dekker es una estrategia para …

- Resolver el problema de la sincronización de procesos.
- Resolver el problema del productor-consumidor.
- Resolver el problema de la exclusión mutua. **X**
- Ninguna de las anteriores.

3) Explique BREVEMENTE los siguientes conceptos:

- Seccion Critica.
- Exclusion Mutua.

4) Evalúe cuidadosamente el siguiente código y responda:

```
0  #include <stdio.h>
1  #include <pthread.h>
2
3  int x = 3;
4
5  void thread_code1() {
6      pthread_exit(0);
7      x = x * 3;
8      printf("La variable x vale %d \n", x);
9  }
10
11 void thread_code2() {
12     x++;
13     printf("La variable x vale %d \n", x);
14     pthread_exit(0);
15 }
16
17 void thread_code3() {
18     x = 5;
19     pthread_exit(0);
20     printf("La variable x vale %d \n", x);
21 }
22
23 int main() {
24     pthread_t hilo1, hilo2, hilo3;
25
26     printf("Hilo Main : %d \n", pthread_self());
27
28     pthread_create(&hilo1, NULL, thread_code3, NULL);
29     pthread_create(&hilo2, NULL, thread_code2, NULL);
30     pthread_create(&hilo3, NULL, thread_code1, NULL);
31
32    //pthread_join(hilo1, NULL);
33    //pthread_join(hilo2, NULL);
34      pthread_join(hilo3, NULL);
35
36 }
```

A) ¿Siempre terminan todos los threads? **¿Por qué?**
No, porque los join para h1 y h2 están comentados.

B) ¿Cuáles son los posibles resultados del programa? **Justifique brevemente**
Hilo Main: 1
-iLa variable x vale 4
La variable x vale 5
La variable x vale 6

C) ¿La función thread_code1 llega a ejecutarse?
Si, pero no hace nada por el pthread_exit(0).

5) Semáforos A y C inicializados en 1, semáforo B en 0. Indicar una secuencia de los mismos que les permitan terminar, y otra que conduzca a un interbloqueo.

Proceso 1	Proceso 2	Proceso 3
Wait (A)	Wait (C)	Wait (B)
Wait (B)	Wait (A)	Wait (A)
	Post (B)	
Post (B)		Post (C)
Post (A)	Post (A)	

Para que terminen: **Todo P2, Todo P1, Todo P3.**
Para que exista Deadlock: **P3 Wait(B) → P2 Wait(C) → P2 Wait(A) → P1 Wait(A) → P2 Wait(B).**