Departamento de Cs. e Ingeniería de la Computación
Universidad Nacional del Sur

Sistemas Operativos y Distribuidos

Trabajo Práctico N° 6

Interbloqueos

Segundo Cuatrimestre de 2015

Ejercicios

1. Para estudiar la problemática asociada a los interbloqueos, se ha propuesto un modelo de sistema el cual consiste en un conjunto de recursos que deben ser distribuidos entre un determinado número de procesos que compiten por los mismos. En este contexto,

   a) ¿Qué entiende por inanición?

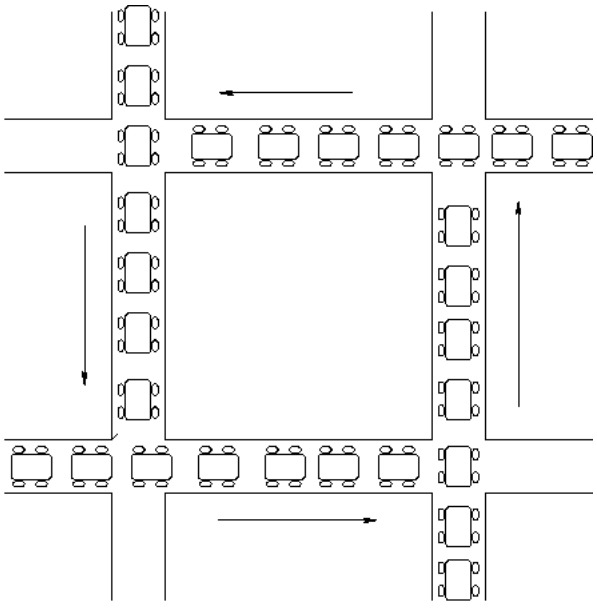
   b) ¿Existe alguna forma de detectar dicha situación?

   c) Proponga al menos una estrategia para mitigar la problemática.

2. Discuta la veracidad de la siguiente afirmación:

   "No hay por qué preocuparse de los interbloqueos, solo pueden ocurrir a nivel de sistema de operativo!"

3. El término *gridlock* generalmente alude a un problema de tráfico en el cual todo vehículo participante es incapaz de moverse.



   a) Muestre que se verifican las cuatro condiciones necesarias para la ocurrencia de un interbloqueo.

   b) Describa una regla simple que permita evitar los interbloqueos en este problema.

4. Suponga que un sistema posee doce recursos, asignados a los procesos P_0, P_1 y P_2 según se indica en la siguiente tabla:

	Max	Mantiene	Necesita
P_0	10	5	5
P_1	4	2	2
P_2	9	3	6

Considerando el estado en el que se encuentra el sistema, determine si los procesos se encuentran interbloqueados.

5. Determine el problema asociado al siguiente programa multi-hilado:

```
#include <pthread.h>
#include <stdio.h>
#include <assert.h>

pthread_mutex_t mutexA;
pthread_mutex_t mutexB;
volatile int shared = 0;

void *routineA (void *arg){
        assert( pthread_mutex_lock(&mutexA) == 0 );
        assert( pthread_mutex_lock(&mutexB) == 0 );

                // Procesamiento sobre shared

        assert( pthread_mutex_unlock(&mutexB) == 0 );
        assert( pthread_mutex_unlock(&mutexA) == 0 );
        pthread_exit(0);
}
void *routineB (void *arg){
        assert( pthread_mutex_lock(&mutexB) == 0 );
        assert( pthread_mutex_lock(&mutexA) == 0 );

                // Procesamiento sobre shared

        assert( pthread_mutex_unlock(&mutexA) == 0 );
        assert( pthread_mutex_unlock(&mutexB) == 0 );
        pthread_exit(0);
}
int main(){
        pthread_t th1, th2;

        pthread_mutex_init(&mutexA, NULL);
        pthread_mutex_init(&mutexB, NULL);

        pthread_create(&th1, NULL, routineA, NULL);
        pthread_create(&th2, NULL, routineB, NULL);

        pthread_join(th1, NULL);
        pthread_join(th2, NULL);

        pthread_mutex_destroy(&mutexA);
        pthread_mutex_destroy(&mutexB);
        return 0;
}
```

6. Proponga una solución al problema planteado en el ejercicio 5 utilizando la primitiva *pthread_mutex_trylock()*. Argumente sobre la política utilizada.

7. Dado el siguiente *snapshot* de un sistema,

	Mantiene			Necesita			Disponible		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	2	0	0	1	1	0			
P_2	3	1	0	0	0	0	0	0	0
P_3	1	3	0	0	0	1			
P_4	0	1	1	0	1	0			

   a) Construya el grafo de asignación de recursos.

   b) Determine si el sistema se encuentra en *deadlock*.

8. Considere la siguiente caracterización de un sistema en un determinado instante de tiempo:

	Asignación				Máximo				Disponible			
	A	B	C	D	A	B	C	D	A	B	C	D
P_1	0	0	1	2	0	0	1	2	1	5	2	0
P_2	1	0	0	0	1	7	5	0				
P_3	1	3	5	4	2	3	5	6				
P_4	0	6	3	2	0	6	5	2				
P_5	0	0	1	4	0	6	5	6				

   a) Escriba la tabla de necesidad.

   b) Determine si el sistema se encuentra en un estado seguro.

   c) Suponga que el proceso P_1 solicita (0,4,2,0). ¿Puede concederse inmediatamente dicha solicitud?

9. Suponga que un sistema posee 150 bloques de memoria, asignados de acuerdo a la siguiente tabla:

	Max	Mantiene
P_0	70	45
P_1	60	40
P_2	60	15

   Aplique el algoritmo del banquero con el objetivo de deteminar si resulta seguro garantizar cada uno de los siguientes requerimientos:

   a) Arriba un cuarto proceso, con una necesidad máxima de 60 bloques de memoria y una necesidad inicial de 25 bloques de memoria.

   b) Arriba un cuarto proceso, con una necesidad máxima de 60 bloques de memoria y una necesidad inicial de 35 bloques de memoria.

10. Evalúe si la implementación del algoritmo del banquero resultaría útil en un sistema operativo real. ***Justifique adecuadamente.***

11. Una posible solución al problema de los filósofos, plantea que todo filósofo hambriento obtenga en primera instancia el palillo que está a su izquierda. Si el palillo que está a su derecha también está disponible entonces el filósofo lo toma y comienza a comer. En caso contrario, el filósofo deja el palillo izquierdo y repite el ciclo.

    Determine el problema asociado a la solución planteada.