Sistemas Operativos II

Practica I

Emiliano Salvatori

Septiembre 2019

1. Practica 1

Interbloqueo

Analice si el siguiente código puede sufrir un potencial interbloqueo suponiendo que se ejecutan en un sistema multiprocesador

Es posible que exista un interbloqueo ya que no se están utilizando las llamadas al sistema que permiten bloquear la utilización del recurso. Estas dos funciones son: *wait()* y *signal*

```
#include <stdio.h>

semaforo recurso_1;
semaforo recurso_2;

void proceso_A(void) {

    down(&recurso_1);
    down(&recurso_2);

    usar_ambos_recursos();

    up(&recurso_2);
    up(&recurso_1);
}

void proceso_B(void) {

    down(&recurso_2);
    down(&recurso_1);

    usar_ambos_recursos();

    up(&recurso_1);
    up(&recurso_2);}
}
```

Indique cómo se puede evitar el potencial interbloqueo

Se puede evitar de la siguiente manera:

```
#include <stdio.h>

semaforo recurso_1;
semaforo recurso_2;

void proceso_A(void) {

    wait(recurso_1);
    wait(recurso_2);

    /*
     * Se comentan las lineas ya que es mejor implementar las llamadas wait() y signal()
    down(&recurso_1);
    down(&recurso_2);
    */

    usar_ambos_recursos();

```

```
19      /*
20          up(&recurso_2);
21          up(&recurso_1);
22      */
23
24      signal(&recurso_2);
25      signal(&recurso_1);
26  }
27
28  void proceso_B(void) {
29
30      wait(recurso_1);
31      wait(recurso_2);
32
33      /*
34      down(&recurso_2);
35      down(&recurso_1);
36      */
37
38      usar_ambos_recursos();
39
40      /*
41      up(&recurso_1);
42      up(&recurso_2);}
43      */
44
45      signal(&recurso_2);
46      signal(&recurso_1);
47  }
```

Indique por qué el siguiente grafo de asignación de recursos tiene un interbloqueo

Se puede observar que existe una situación de interbloqueo debido a que se dan las 4 condiciones para ello, a saber:

- **Exlusión Mutua**: Es decir que hay *recursos no compartidos* de uso exlusivo para cada proceso. El proceso toma para sí un recurso o una instancia de ese recurso.
- **Contención y espera**: Un proceso toma para sí un recurso o una instancia de ese recurso y solicita y espera otro recurso que solicitó.
- **Condición no apropiativa**: No se le puede expropiar un recurso otorgado a un proceso, sino que debe liberarlo por sí solo, no pueden ser quitados por la fuerza.
- **Espera circular**: Efectivamente se puede observar en el grafo que se hayan los procesos en una espera circular entre ellos y los recursos solicitados/obtenidos. Cada uno de los procesos espera el siguiente recurso que a su vez está asignado a otro proceso. Esto genera que haya una interdependencia.

En el grafico se puede observar que existe un ciclo donde cada proceso tiene asignado un determinado recurso que a su vez está siendo solicitado por otro proceso. De esta forma, como el recurso no puede serle expropiado a los procesos de una forma deliberada, sólo podrá liberarlo cuando el proceso haya terminado de utilizarlo. El problema que se da es que se espera una situación que nunca se dará, ya que ninguno de los procesos podrá liberar el recurso obtenido ni tampoco podrá obtener el siguiente recurso para terminar su tarea.

Implemente el algoritmo de detección de Deadlock teniendo en cuenta que se tienen 3 procesos y 4 tipos diferentes de recursos (Unidades de CD-ROM, Escaneres, Impresoras y lectoras SD). En este contexto el proceso 1 tiene un escáner, el proceso 2 tiene dos lectoras SD y una unidad de CD-ROM y el proceso 3 tiene una impresora y dos escáneres

Tabla de Requerimientos por proceso:

-	SD	Impresora	Escáner	Cd-Rom
P1	2	0	0	1
P2	1	0	3	0
P3	2	1	0	0

Tabla de recursos disponibles:

SD	Impresora	Escáner	Cd-Rom
4	2	3	1

Para ello lo que se realiza es una tabla de asignación, donde se deja asentado cómo se asignan los recursos siguiendo el *método del banquero*:

Asignación en **Iteración nº 1**:

-	SD	Impresora	Escáner	Cd-Rom
P1	2	0	0	1
P2	1	0	3	0

Requerimientos en **Iteración nº 1**:

-	SD	Impresora	Escáner	Cd-Rom
P1	0	0	0	0
P2	0	0	0	0
P3	2	1	0	0

Recursos disponibles en **Iteración nº 1**:

SD	Impresora	Escáner	Cd-Rom
1	2	0	0

Luego de esta primera iteración el proceso nº 1 y el nº 2 terminan, quedando la tabla de Recursos disponibles de la siguiente manera:

Recursos disponibles:

SD	Impresora	Escáner	Cd-Rom
4	2	3	1

Ahora es posible brindarle los recursos al proceso que falta:

Asignación en **Iteración nº 2**:

-	SD	Impresora	Escáner	Cd-Rom
P3	2	1	0	0

Requerimientos en **Iteración nº 2**:

-	SD	Impresora	Escáner	Cd-Rom
P1	0	0	0	0
P2	0	0	0	0
P3	0	0	0	0

Recursos disponibles en **Iteración nº 2**:

SD	Impresora	Escáner	Cd-Rom
2	1	3	1

Una vez finalizado este segundo ciclo, el proceso nº 3 finaliza por lo que se obtiene la siguiente tabla de Recursos Disponibles:

Tabla de recursos disponibles luego de la **Iteración nº 2**:

SD	Impresora	Escáner	Cd-Rom
4	2	3	1

Indique por qué el siguiente grafo de asignación de recursos tiene un interbloqueo