TAREA

Entrega: Viernes 26/11/2004

El problema de la cena de los filósofos

1. Había una vez cinco filósofos que vivían juntos. La vida de cada filósofo consistía principalmente en pensar y comer y, tras años de pensar, todos los filósofos se habían puesto de acuerdo en que la única comida que contribuía a sus esfuerzos era el arroz.
   Los preparativos de la comida eran simples: una mesa redonda en la que había cinco platos de arroz, uno para cada filósofo, y cinco palillos. Un filósofo que quisiera comer iría a su lugar asignado en la mesa y, usando los dos palillos de cada lado del plato, se comería el arroz. El problema es el siguiente: inventar un ritual (algoritmo) que permita comer a los filósofos. El algoritmo debe satisfacer la exclusión mutua (dos filósofos no pueden emplear el mismo palillo a la vez), además de evitar el interbloqueo y la inanición (en este caso, este ultimo término tiene un significado literal además del algorítmico).

   Una primera solución al problema de la cena de los filósofos es:

```
/* programa cena_filósofos */
semáforo palillo[5] = {1};
int i;
void filosofo(int i)
{
        while (cierto)
        {
                pensar ( );
                wait (palillo [i]);
                wait (palillo [(i + 1) mod 5];
                comer ( );
                signal (palillo [(i + 1) mod 5]);
                signal (palillo [i]);
        }
}
void main ( )
{
        parbegin (filosofo (0), filosofo (1), filosofo (2), filosofo (3), filosofo (4));
}
```

Este algoritmo sugiere una solución por medio de semáforos. Cada filosofo toma primero el palillo de su izquierda, y después el de su derecha. Cuando un filosofo termina de comer, devuelve los dos palillos a la mesa. Esta solución, desafortunadamente, produce ínterbloqueo: si todos los filósofos están hambrientos al mismo tiempo, todos se sientan, todos toman el palillo de su izquierda y todos intentan tomar el otro palillo que no estará. Esta solución es poco decorosa, pues todos los filósofos pasan hambre.
Para superar el riesgo de ínterbloqueo se podrían adquirir 5 palillos adicionales (una solución mas saludable), o enseñar a los filósofos a comer arroz con un solo palillo. Su misión: proponer una modificación a este algoritmo libre de interbloqueo e inanición.

Nota: No hagan caso de la solución de los libros, pues alguna de ellas puede conducir a inanición.

2. Cenicienta y el Príncipe se quieren divorciar. Para dividir sus propiedades, han acordado el siguiente algoritmo. Cada mañana uno de ellos debe enviar una carta al abogado del otro para solicitar un elemento de su propiedad. Puesto que una carta tarda un día en ser entregada, han acordado que si ambos descubren que han solicitado el mismo artículo el mismo día, al día siguiente enviarán una carta para cancelar la solicitud. Entre sus propiedades están su perro Woofer, su perrera, su canario Piolín y la jaula del canario. Los animales aman sus casas, por lo cual han acordado invalidar cualquier división de la propiedad que separe a un animal de su casa, por lo que la división deberá volver a iniciarse desde cero. Tanto Cenicienta como el príncipe desean de forma desesperada a Woofer. Ambos se van de vacaciones (separados) y cada uno de ellos programa una computadora personal para manejar la negociación. Al regresar de sus vacaciones, las computadoras continúan negociando. ¿Por qué? ¿Es posible el bloqueo? ¿Es posible la inanición? Explique.

Resolución:

1.- Cena de los filósofos:

Existen varios algoritmos que permiten reducir la probabilidad de ocurrencia de interbloqueo, y otros que la eliminan por completo. En todos los casos los palillos son representados con objetos de exclusión mutua. En el primer tipo de algoritmo tenemos el siguiente: Cada proceso (filosofo) toma dos palillos (wait del semáforo), uno a la vez pero el orden de cual palillo toma primero (izquierdo o derecho) se determina de forma aleatoria dentro del proceso. De esta manera se reduce, mas no se elimina la posibilidad de interbloqueo.

Una variante del algoritmo anterior es aquel donde todos los procesos esperan por los semáforos en un orden especifico e igual para todos, salvo un proceso, donde el orden de petición de los palillos (semáforos) es el inverso. De esta manera se evita el interbloqueo. La solución que se mostrara a continuación corresponde a otro algoritmo donde se utiliza la instrucción **wait** de dos objetos simultáneamente. Con este algoritmo se elimina por completo el problema de interbloqueo e inanición por completo. La explicación aparece dentro del mismo código.

NOTA: el **wait** aquí descrito pone en espera al proceso hasta que los 2 argumentos (semáforos) estén señalizados

```
/* Programa cena de los filósofos. Sin interbloqueo */
#define N 5                     //N = numero de comensales, palillos, filósofos, etc…
semáforo palillo[N]={1};        //arreglo de N semáforos que representan los palillos. Todos en 1

void filosofo (int i)           //proceso de cada filosofo
{
    while (true)        //bucle infinito, se puede colocar condición de parada con variable
    {
        pensar ( );                         //tiempo de espera aleatorio de c/filosofo antes de 'comer'
        wait (palillo [i], palillo[i +1] mod 5); //pido los 2 palillos simultáneamente
        comer ( );                          //tiempo de espera mientras come (puede ser aleatorio)
        signal (palillo [i]);               //libero palillo izquierdo
        signal (palillo [(i + 1) mod 5]);   //libero palillo derecho

        /* El orden en que libero los palillos no es de especial relevancia. */
    }
}

void main()
{
    parbegin (filosofo (0), filosofo (1), filosofo (2), filosofo (3), …, filosofo (N));
    //inicio de los procesos (filósofos)
}
```

2.- Partición de bienes:

Tanto Cenicienta como el Príncipe se pueden asociar con dos procesos de igual prioridad que hacen petición de un conjunto de recursos (objetos de exclusión mutua) que representan el perro, la perrera, el pájaro y su jaula. Las peticiones que se efectúan por carta, son totalmente sincronías, por lo cual no hay forma de determinar la prioridad de la petición según el orden de llegada. Como ambos procesos piden primero al perro (desean de forma desesperada a Woofer)

Al efectuar ambos la petición del perro como primera petición, dicha solicitud será cancelada al día siguiente como se acordó en un principio. En caso de que se asigne el perro a cualquiera de los dos entes (procesos), existe la posibilidad de que el otro proceso se adjudique la casa del mismo. Esta transacción se anulara también, y se comenzara desde cero puesto que se acordó que no será valida ninguna partición donde algún animal quede sin su casa. Al reiniciarse el algoritmo, si no se usa un elemento aleatorio, existe el riesgo de interbloqueo, y por ende de inanición.

Una posible solución es que las peticiones no sean síncronas, o que exista un manejo de las prioridades de los procesos por parte del abogado (Sistema Operativo) para la asignación de los recursos.