Programación Concurrente – 2013
Práctica Nº2 – Plan 2003 y 2007

Semáforos

Nota: En todos los ejercicios el tiempo debe representarse con la función delay

CONSIDERACIONES PARA RESOLVER LOS EJERCICIOS:

1- **Se pueden utilizar semáforos binarios (valor 0-1).**
2- **Se pueden utilizar semáforos n-arios (valor 0-n).**
3- **Los semáforos deben estar declarados en todos los ejercicios.**
4- **Los semáforos deben estar inicializados en todos los ejercicios.**
5- **No se puede utilizar ninguna operación que determine el valor de un semáforo.**
6- **Debe evitarse el busy waiting en todos los ejercicios.**
7- **Siempre debe maximizarse la concurrencia (sin contradecir el enunciado)**

1. Existen N personas que deben ser chequeadas por un detector de metales antes de poder ingresar al avión. Implemente una solución que modele solo el acceso de la persona al detector (es decir si el detector esta libre la persona lo puede utilizar caso contrario debe esperar).

   Modifique su solución para que funcione en el caso que el detector pueda controlar a tres personas a la vez.

2. Suponga que existe una BD que puede ser accedida por 6 usuarios como máximo al mismo tiempo. Además los usuarios se clasifican como usuarios de prioridad alta y usuarios de prioridad baja. Por último la BD tiene la siguiente restricción:
   - no puede haber más de 4 usuarios con prioridad alta al mismo tiempo usando la BD
   - no puede haber más de 5 usuarios con prioridad baja al mismo tiempo usando la BD

```
Var
 sem s = 6;
 sem alata= 4;
 sem baja = 5;
```

Palta [I:1..L]::	**Pbaja [I:1..K]**
{ P (s); P(alta); <usa la BD> V(s); V(alta); }	{ P (s); P(baja); <usa la BD> V(s); V(baja); }

   La solución presentada es la mejor?. Justifique.

3. Implemente en semáforos el problema de productor-consumidor descripto a continuación. Existe un proceso productor que realiza productos indefinidamente. Cada vez que termina de producir un producto lo guarda en un buffer que tiene una capacidad de un elemento. Además existen C consumidores que intentan consumir los productos producidos por el productor.
   Cada vez que un producto es consumido por los C consumidores el productor puede depositar un nuevo producto.

4. Suponga que se tiene un curso con 50 alumnos. Cada alumno elije una de las 10 tareas para realizar entre todos. Una vez que todos los alumnos eligieron su tarea comienzan a realizarla. Cada vez que un alumno termina su tarea le avisa al profesor y si todos los alumnos que tenían la misma tarea terminaron el profesor les otorga un puntaje que representa el orden en que se terminó esa tarea.
   Nota: Para elegir la tarea suponga que existe una función ***elegir*** que le asigna una tarea a un alumno (esta función asignará 10 tareas diferentes entre 50 alumnos, es decir, que 5 alumnos tendrán la tarea 1, otros 5 la tarea 2 y así sucesivamente para las 10 tareas). El tiempo en un alumno tarda en realizar la tarea es random.

5. Existen N alumnos que desean consultar a alguno de los A ayudantes en una práctica. Para esto cada alumno se agrega en una cola para consultas. Si una vez que el alumno se agregó en la cola pasaron más de 15 minutos el mismo se retira; por el contrario si fue atendido por un ayudante, entonces el alumno le entrega su ejercicio resuelto y espera la opinión del ayudante. Si el ejercicio estaba correcto el alumno se retira, en caso que tenga que modificarlo, realiza las modificaciones y vuelve a agregarse a la cola (Ya no debe tenerse en cuenta lo de los 15 minutos).
   Nota: Suponga que existe una función que devuelve si un alumno hizo bien o mal el ejercicio. El alumno no decide a cual ayudante le consulta. Los 15 minutos solo deben tenerse en cuenta la primera vez, es decir, si el alumno fue atendido ya no se toma más en cuenta el tiempo.

6. Suponga que hay N tareas que se realizan en forma diaria por los operarios de una fábrica. Suponga que existen M operarios (M = Nx5). Cada tarea se realiza de a grupos de 5 operarios, ni bien llegan a la fábrica se juntan de a 5 en el orden en que llegaron y cuando se ha formado el grupo se le da la tarea correspondiente empezando de la tarea uno hasta la enésima.
   Una vez que los operarios del grupo tienen la tarea asignada producen elementos hasta que hayan realizado exactamente X entre los operarios del grupo. Una vez que terminaron de producir los X elementos, se juntan los 5 operarios del grupo y se retiran.
   Tenga en cuenta que pueden salir varios grupos de operarios al mismo tiempo y que no debe interferir con la entrada.
   Nota: cada operario puede hacer 0, 1 o más elementos de una tarea. El tiempo que cada operario tarda en hacer cada elemento es diferente y random. Maximice la concurrencia.

7. Resolver el siguiente problema. Existen N camiones que deben descargar cereales en una acopiadora. Los camiones se descargan de a uno por vez, y de acuerdo al orden de llegada. Una vez que el camión llegó, espera a lo sumo 2 hs. a que le llegue su turno para comenzar a descargar su cereal, sino se retira sin realizar la descarga.
   Cuando un camión termina de realizar la descarga de su contenido, se debe encargar de avisarle al siguiente camión (en caso de que haya alguno esperando) que le llegó su turno para descargar.

8. En un curso hay dos profesores que toman examen en forma oral, el profesor A llama a los alumnos de acuerdo al orden de llegada, mientras que el profesor B llama a cualquier alumno (que haya llegado). Existen N alumnos que llegan y se quedan esperando hasta ser llamados para rendir, luego de que uno de los dos profesores lo atiende, se va. Indicar si la siguiente solución realizada con semáforo resuelve lo pedido. Justificar la respuesta.

```
string estado[N] = ([N], “Esperando” )
queue colaA, colaB
sem llegoA, llegoB = 0
sem esperando[N] = ([N], 0)
sem mutex[N] = ([N], 1)
sem mutexA, mutexB = 1
```

Profesor A::

```
{ int  idAlumno
  while (true)
   { P(llegoA)
     P(mutexA)
     idAlumno = pop(colaA)
     V(mutexA)
     P(mutex[idAlumno])
     If (estado[idAlumno] = =“Esperando”)
        estado[idAlumno] = “A”
        V(mutex[idAlumno])
        V(esperando[idAlumno])
        //Se toma el examen//
        V(esperando[idAlumno])
     else
        V(mutex[idAlumno])
  }
}
```

Profesor B::

```
{ int idAlumno
  while (true)
   { P(llegoB)
     P(mutexB)
     idAlumno = popAleatorio(colaB)
     V(mutex(B))
     P(mutex[idAlumno])
     If (estado[idAlumno] == “Esperando”)
        estado[idAlumno] = “B”
        V(mutex[idAlumno])
        V(esperando[idAlumno])
        //Se toma el examen//
        V(esperando[idAlumno])
     else
        V(mutex[idAlumno])
  }
}
```

Alumno[i: 1..N]

```
{ P(mutexA)
  push(colaA, i)
  V(mutexA)
  P(mutexB)
  push(colaB, i)
  V(mutexB)
  P(esperando[i])
  if (estado[i] == “A”)
     //Interactúa con el Prof A//
  else
     //Interactua con el Prof B//
  P(esperando[i])
}
```