Programación Concurrente

Práctica 5: Semáforos (cont.)

Ejercicio 1. Se busca computar la suma de los primeros N números impares utilizando dos threads llamados `generador` y `acumulador`, que deben cooperar entre sí para resolver esta tarea. Contamos con dos variables globales `impar`, que comienza inicializada en N, y `suma`, que comienza inicializada en 0. Buscamos que `generador` se encargue de que `impar` contenga el valor i cuando se deba sumar el i-ésimo número impar, y que `acumulador` compute cuál es el i-ésimo impar y lo sume a `suma`. Al finalizar el cómputo, el thread `generador` debe imprimir el valor correcto de la sumatoria.

a) Escriba un programa concurrente que funcione según esta dinámica.

b) ¿Qué modificación debería hacerse si existieran múltiples threads `generador`? ¿Y si hubieran múltiples threads `acumulador`? ¿Existe posibilidad de *starvation*?

Ejercicio 2. Para cruzar un determinado río, se tiene un pequeño bote que lleva a la gente que lo necesite de una costa a la otra. El bote tiene capacidad para una persona y sólo hace el recorrido en una dirección (en una primera instancia, el viaje de regreso no lleva pasajeros). Las personas se van acercando a la orilla de partida y esperan poder subir al bote, en orden de llegada. Aquella que lo logra, se sube y se acomoda en el asiento, lo cual toma unos minutos. Una vez acomodada la persona, el bote sale hacia la costa opuesta. Durante ese interín, la persona a bordo debe esperar. Al llegar, la persona debe descender (que, de nuevo, toma algo de tiempo), y una vez que terminó de bajar, el bote regresa a la costa original para tomar a la siguiente persona.

a) Modele el escenario descrito utilizando un thread `transbordador` para el bote y uno `persona` para cada persona que llegue a la costa. Modele las esperas a realizar mediante el uso de semáforos.

b) Suponga ahora que se cuenta con un transbordador con capacidad para N personas, y que en esta oportunidad el barco lleva gente tanto en el viaje de ida como en el de regreso, teniendo la siguiente dinámica.

   - Empieza en la costa *oeste* (podemos pensar que es la costa "0").
   - Espera en una costa hasta que haya gente acomodada en los N asientos.
   - Viaja hasta la costa *este* ("1").
   - Amarra, permitiendo que los pasajeros desciendan.
   - Repite el procedimiento desde el principio, en la costa actual.

   Tenga en cuenta que el thread `pasajero` puede llevar un parámetro que indica en qué costa comienza su viaje.

   Se proponen dos variantes:

   (i) Cuando el bote llega a una costa, espera que todos sus ocupantes terminen de bajar antes de permitir subir a la gente que está esperando allí.

(ii) Cuando el bote llega a una cosa, la gente empieza a bajar y subir en forma concurrente, sólo se busca que no hayan más de N personas sobre el barco en un momento dado.

Ejercicio 3. En un gimnasio hay cuatro aparatos, cada uno para trabajar un grupo muscular distinto. Los aparatos son cargados con discos (el gimnasio cuenta con 20 discos, todos del mismo peso). Cada cliente del gimnasio posee una rutina que le indica qué aparatos usar, en qué orden y cuanto peso utilizar en cada caso (asuma que la rutina es una lista de tuplas con el número de aparato a usar y la cantidad de discos cargar, la rutina podría incluir repeticiones de un mismo aparato). Como norma el gimnasio exige que cada vez que un cliente termina de utilizar un aparato descargue todos los discos y los coloque en el lugar destinado a su almacenamiento (lo que incluye usos consecutivos del mismo aparato).

a) Indique cuales son los recursos compartidos y roles activos.

b) Escriba un código que simule el funcionamiento del gimnasio, garantizando exclusión mutua en el acceso a los recursos compartidos y que esté libre de deadlock y livelock.

Ayuda: Considere modelar a los clientes como threads (cada uno con su propia rutina) y a los aparatos como un arreglo de semáforos.

Ejercicio 4. En un famoso shopping de la ciudad se agregó un puesto de tintorería automática para que la gente pueda dejar su ropa mientras recorre los locales. Novedosamente, estas máquinas entregan un beeper a la gente que parpadea para indicar que la ropa ya está lista para ser retirada. Hay K máquinas de lavado listas para funcionar. Cada persona que quiere utilizar el servicio (en este ejemplo no modelaremos a quieres no lo requieran) espera que haya alguna máquina disponible para serle asignada según disponibilidad, carga su ropa y recibe su beeper. Luego, se va a mirar locales, y si al terminar su beeper aún no parpadea, espera que lo haga para volver al local y retirar sus prendas (Si el beeper parpadea mientras la persona aún esta mirando locales, esta lo ignora hasta que está por retirarse). Modele este escenario por medio de semáforos. Procure que no suceda que una persona vaya a retirar su ropa cuando se terminó de procesar la de alguien más.

Ejercicio 5. Considere el ejercicio 2 y piense este escenario adicional:

a) Modifique la solución de la variante (i) considerando que ahora que hay K botes distintos llevando gente de manera concurrente (donde cada uno viaja a su tiempo). Todos los barcos comienzan en la costa 0. El thread `transbordador` recibe como parámetro un id de barco único. Modele el problema prestando atención a que el comportamiento sea consistente (ie. que no pase que una persona se suba a un bote y se baje de otro).

Ejercicio 6. Se avecina el partido super-mega clásico entre dos equipos que llamaremos BJ y RP. Para evitar conflictos en la cancha, se dispuso el siguiente mecanismo de control de acceso: No se permitirá que la diferencia entre la gente de la hinchada de BJ y la de gente de la hinchada RP sea mayor que 1 (toda persona que va a ver el partido es de alguno de los dos equipos). Cada persona que llegue a la cancha debe asegurarse de esta regla le permite el acceso. Si no, esperará hasta que pase alguien del equipo opuesto de manera de conseguir acceso. Una vez que una persona entró a la cancha, se quedará hasta finalizar el partido (que no modelaremos,

estipulando que este sigue indefinidamente).

a) Modele este comportamiento utilizando Semáforos.

b) Extienda la solución anterior contemplando que la cancha tiene una capacidad de N personas. Cada persona que logra entrar a la cancha ocupa un lugar, y como estos nunca se liberan, en algún momento la cancha estará llena. En ese momento, toda persona que llegue deberá retirarse sin tener la posibilidad de ingreso. Asegúrese, también, que ninguna persona se quede esperando indefinidamente.

c) El partido ya se jugó y resultó uno de los más *épicos* de la historia. Por ello, se planificó proyectarlo en el cine de la ciudad para que la gente pueda verlo en pantalla gigante. La gente que desea asistir sólo debe ir al cine y retirar una entrada, disponible en un mostrador del lobby (suponemos que las personas las toman de a una, aunque es posible que dos personas se acerquen a buscar su entrada en momentos muy cercanos. No hay distinción entre personas de un equipo y otro). La cantidad total de entradas es M, y si alguien llega luego de que se hayan agotado, se retirará del cine. En un horario determinado, el personal del cine quita las entradas del mostrador y habilita el acceso a la sala las personas que tengan su entrada. Luego, se espera a que todas las personas se hayan acomodado en alguna butaca, el personal cierra las puertas, y comienza la proyección. Como en el caso anterior, podemos pensar que la gente que entró se queda mirando el partido indefinidamente. Modele este nuevo escenario utilizando Semáforos. Preste atención a que el personal del cine no se quede esperando que se sienten más personas que las que las que retiraron entradas.

Ejercicio 7. Dada la solución al problema de Lectores-Escritores con prioridad para escritores, considere las siguientes modificaciones al thread `Lector`:

a) ¿Qué efecto produciría si se usara un `permisoL` para todo el cuerpo del Lector? Es decir, con el siguiente código para los lectores:

```
thread Lector() {
  mutexP.acquire();
  permisoL.acquire();
  lectores++;
  if (lectores == 1)
    permisoE.acquire();

  leer();

  lectores--;
  if (lectores == 0)
    permisoE.release();
  permisoL.release();
  mutexP.release();
}
```

b) ¿Funcionaría la solución si se quitara el semáforo `mutexL` delegando la exclusión mutua de los lectores a `permisoL`? Es decir, con el siguiente código para los lectores:

```
thread Lector: {
  mutexP.acquire();
```

```
    permisoL.acquire();
    lectores++;
    if (lectores == 1)
       permisoE.acquire();
    permisoL.release();
    mutexP.release();

    leer();

    permisoL.acquire();
    lectores--;
    if (lectores == 0)
      permisoE.release();
    permisoL.release();
  }
```

Ejercicio 8. En una oficina hay un baño unisex con 8 toiletes. A lo largo del día, distintas personas entran a utilizarlo. Si sucede que en ese momento todos los toiletes están ocupados, las personas esperan hasta que alguno se libere. Por otra parte, periódicamente el personal de limpieza debe pasar a mantener las instalaciones en condiciones. La limpieza del baño no se puede hacer mientras haya gente dentro del mismo, por lo que si en ese momento hay personas utilizando algún toilete o esperando que se libere alguno, el personal de limpieza debe esperar a que el baño se vacíe completamente. En contraparte, si hay un empleado de limpieza trabajando en el baño, las personas que quieran utilizarlo deberán esperar a que termine.

a) Modele esta situación utilizando semáforos como mecanismo de sincronización (puede modelar al personal de limpieza como un único thread).

b) Modifique la solución anterior para contemplar el caso donde el personal de limpieza tiene prioridad. Es decir, si hay un empleado de limpieza esperando para hacer el mantenimiento, las siguientes personas que lleguen deben esperar a que logre terminar la limpieza.

Ejercicio 9. Se desea implementar un sistema de control para una estación de servicio con 6 puestos de carga. El sistema debe garantizar que en ningún momento puede haber más de 6 vehículos cargando combustible y que la atención se produce en orden de llegada. Además, la estación cuenta con 1 puesto para el abastecimiento de combustible que es provisto por camiones que arriban con mucha menor frecuencia que los clientes. Por lo tanto se requiere garantizar que no más de un camión pueda descargar combustible al mismo tiempo.

a) Identifique los roles activos y los recursos compartidos.

b) De una solución considerando que el abastecimiento de combustible no puede hacerse al mismo tiempo que la carga a clientes y que los camiones tienen prioridad por sobre los vehículos. ¿Es su solución libre de inanición?

Ejercicio 10. Se desea modelar el control de tránsito de un puente que conecta dos ciudades. Dado que el puente es muy estrecho se debe evitar que dos autos circulen al mismo tiempo en

dirección opuesta, dado que quedarían atascados.

Resuelva los siguientes problemas usando semáforos, modelando cada coche como un thread independiente que desea atravesar el puente en alguna de las dos direcciones posibles. Tenga en cuenta que atravesar el puente no es una acción atómica, y por lo tanto, requiere de cierto tiempo.

a) De una solución que permita que varios coches que se desplazan en la misma dirección puedan circular simultáneamente.

b) Modifique la solución anterior para que como máximo 3 coches puedan circular por el puente al mismo tiempo.

c) Indique si la solución propuesta en el punto b es libre de inanición. Justifique su respuesta.