CC4302 Sistemas Operativos – Control 1 – Semestre Otoño 2017 – Prof.: Luis Mateu

Pregunta 1

Un *left/right lock* es un mutex que se puede otorgar completo o por mitades. Solo un thread puede obtener el mutex completo llamando a *fullLock*. Posteriormente, ese thread devuelve el mutex con *fullUnlock*. Hasta 2 threads pueden obtener cada uno una mitad distinta del mutex por medio de *halfLock*, que entrega LEFT si se otorgó la mitad izquierda o RIGHT si fue la mitad derecha. Más tarde cada thread devuelve su respectiva mitad con *halfUnlock*, especificando cuál fue la mitad que se le había otorgado. La siguiente implementación es incorrecta e ineficiente, pero funciona casi siempre:

```
enum { LEFT= 0, RIGHT= 1 };
int busy[2]= { FALSE, FALSE }; // ambas mitades libres
nSem m; // = nMakeSem(1); parte con 1 ticket
```

```
int halfLock() {
  if (!busy[LEFT] &&
      !busy[RIGHT])
    nWaitSem(m);
  int l= -1;
  while (l==-1) {
    if (!busy[LEFT])
      l= LEFT;
    else if (!busy[RIGHT])
      l= RIGHT;
  } // ¡busy waiting!
  busy[l]= TRUE;
  return l;
}
```

```
void halfUnlock(int l) {
  busy[l]= FALSE;
  if (!busy[LEFT] &&
      !busy[RIGHT])
    nSignalSem(m);
}
void fullLock() {
  nWaitSem(m);
}
void fullUnlock() {
  nSignalSem(m);
}
```

a) (1.5 puntos) Haga un diagrama de threads que muestre que estando el mutex completamente libre, puede ocurrir que 2 threads invoquen concurrentemente *halfLock* pero solo el primero obtenga una mitad. El segundo espera hasta que el primero invoque *halfUnlock*.

b) (1.5 puntos) Haga un diagrama de threads que muestre que se puede otorgar erróneamente medio mutex y el mutex completo simultáneamente. *Ayuda:* habiendo un thread obtenido una mitad del mutex con *halfLock*, cuando éste invoca *halfUnlock* concurrentemente con un 2do thread que invoca *halfLock*, el semáforo *m* puede quedar con un 1 ticket. Así un 3er thread puede obtener el mutex completo con *fullLock*, mientras que el 2do thread ya posee una mitad.

c) (3 puntos) Reeimplemente correctamente y sin *busy-waiting* las funciones *halfLock* y *halfUnlock*, ***sin modificar las funciones fullLock y fullUnlock***. Para ello preserve la idea de la solución de más arriba, manteniendo el semáforo *m*, pero agregando 2 nuevos semáforos de nSystem. El primer semáforo para lograr la exclusión mutua dentro de *halfLock* y dentro de *halfUnlock*. El segundo semáforo mantiene hasta 2 tickets, uno por cada mitad libre del mutex. No se preocupe por la hambruna.

Pregunta 2

La municipalidad de Geek Island posee una barcaza para el transbordo de vehículos desde la isla hacia el continente o viceversa. La barcaza tiene capacidad para llevar un solo vehículo. Los conductores son representado por tareas de nSystem. Un conductor con su vehículo situado en el embarcadero del continente puede solicitar cruzar hacia la isla invocando la función *cruzarAIsla(v)* en donde *v* es el vehículo de ese conductor. Del mismo modo se invoca *cruzarACont(v)* cuando el vehículo se sitúa en el embarcadero de la isla para cruzar hacia el continente. La siguiente es una implementación ineficiente de estas 2 funciones:

```
nSem mutex; // = nMakeSem(1);
int enIsla= FALSE; Barcaza *b; // = nuevaBarcaza();
```

```
void cruzarAIsla(Vehiculo *v) {
  nWaitSem(mutex);
  if (enIsla)
    navegarACont(b);
  embarcar(v, b);
  navegarAIsla(b);
  desembarcar(v, b);
  enIsla= TRUE;
  nSignalSem(mutex);
}
```

```
void cruzarACont(Vehiculo *v) {
  nWaitSem(mutex);
  if (!enIsla)
    navegarAIsla(b);
  embarcar(v, b);
  navegarACont(b);
  desembarcar(v, b);
  enIsla= FALSE;
  nSignalSem(mutex);
}
```

En donde *navegarACont*, *navegarAIsla*, *embarcar* y *desembarcar* son funciones dadas y las 2 primeras demoran un tiempo considerable. Esta implementación no es eficiente porque la barcaza podría navegar vacía hacia la isla aún cuando existe un vehículo en espera en el continente (y viceversa).

Programe una solución eficiente de *cruzarAIsla* y *cruzarACont* usando los monitores de nSystem. Considere las siguientes restrucciones: (i) la barcaza no puede navegar hacia la isla si no se encuentra detenida en el continente, y viceversa, (ii) las 2 funciones pedidas solo pueden retornar una vez que la barcaza está detenida en la orilla de destino y el vehículo ha sido desembarcado, (iii) si la barcaza navega hacia el continente y hay vehículos en espera en la isla, Ud. debe embarcar un y solo un vehículo en la isla y debe corresponder al que lleva más tiempo esperando, (iv) la barcaza no debe permanecer detenida si hay vehículos en espera en alguna de las orillas, y (v) Ud. sí debe detener la barcaza si no hay vehículos que transportar en ninguna de las orillas.

Ayuda: Use una cola por cada orilla para encolar las solicitudes de traslado. Cree una tarea adicional que da las órdenes a la barcaza. Esta tarea embarca un vehículo de la cola del continente (si existe), navega a la isla, lo desembarca, embarca un vehículo de la cola de la isla, navega al continente, lo desembarca, y así sucesivamente. Si no hay vehículos en ninguna de las orillas, se espera con la barcaza detenida, hasta que llegue uno a cualquiera de las 2 orillas.