Clase práctica - Problemas de Sincronización

Sistemas Operativos - 1c 2012

Repaso de Semáforos

Qué era un semáforo?

- NO es un entero
- Una variable que provee contención sobre la ejecución de un pedazo de código
- Se inicializa al crearse:
  `Semaphore sem = Semaphore(x)`
  con un valor inicial `x` (generalmente no negativo)
- Se accede a través de dos primitivas:
  `sem.wait()`
  `sem.signal()`

Secuencia - Problema

Problema:

- Tenemos una serie de procesos $P_i, i \in \{1 \dots N\}$ que se están ejecutando en simultáneo (supongamos que tenemos N CPUs).
- Cada proceso i tiene una sentencia s_i.
- Queremos que en el sistema global se ejecuten en orden $s_1, s_2, \dots, s_N$.

Secuencia - Solución (inicialización)

```
void init()
{
  semaforos = Semaphore[N+1]

  for (i = 0; i < N+1; i++)
    semaforos[i] = new Semaphore(0)

  for (i = 0; i < N; i++)
    fork P(i)

  semaforos[0].signal()
}
```

Secuencia - Solución (procesos)

```
void P(i)
{
  semaforos[i].wait()
  s(i)
  semaforos[i+1].signal()
}
```

Rendezvous - Problema

Problema:

- Dados dos procesos $P_1 = [a(1); b(1)]$ y $P_2 = [a(2); b(2)]$
- Los $b(i)$ deben ejecutarse *después* de que hayan ejecutado los $a(i)$.
- Sin embargo, no hay que restringir de más: no hay que imponer ningún orden entre los $a(i)$ ni entre los $b(i)$.

Rendezvous - Solución

Variables:

```
Semaphore terminoA1 = Semaphore(0);
Semaphore terminoA2 = Semaphore(0);
```

Solución:

```
P1:                          P2:
  a1                           a2
  terminoA1.signal()           terminoA2.signal()
  terminoA2.wait()             terminoA1.wait()
  b1                           b2
```

Barrera - Problema

Problema:

- El mismo problema de antes pero con una cantidad arbitraria de procesos $P_i = [a(i); b(i)]$.

Barrera - Solución

Variables:

```
Semaphore mutex = Semaphore(1)
int cuantosLlegaron = 0
Semaphore barrera = Semaphore(1)
```

Solución:

```
P(i):
  a(i)
  mutex.wait()
   cuantosLlegaron++
  mutex.signal()
  if (cuantosLlegaron == N)
    barrera.signal()
    barrera.wait()
    b(i)
```

Cola FIFO - Problema

Problema:

- Una cantidad arbitraria de procesos P_i.
- Son lanzados en momentos arbitrarios.
- Deben ser atendidos de a uno, en orden de llegada.

Cola FIFO - Solución (variables)

Variables globales:

```
Semaphore mutex = Semaphore(1)
Cola<Semaphore> cola = Cola<Semaphore>()
```

Variables locales de cada P_i:

```
Semaphore sem = Semaphore(0);
```

Cola FIFO - Solución (procesos)

```
P(i):
  mutex.wait()
   cola.agregar(sem)
   if (cola.size == 1)
     sem.signal()
  mutex.signal()
  sem.wait()

  ...

  mutex.wait()
   cola.pop()
   if (cola.size > 0)
     cola.prim.signal()
  mutex.signal()
```

Papá Noel - Problema

El problema de Papá Noel (no el del arca), los renos vagos y los elfos molestos.

Por el gran trabajo que implica repartir regalos a todos los chicos y chicas del mundo en una sola noche (aunque seguro que aprovecha que la tierra gira), Papá Noel duerme todo el resto del año en su cabaña en el Polo Norte.

En ese tiempo, los 9 renos que transportan a Papá Noel se toman vacaciones en alguna isla del Pacífico Sur. Mientras tanto, los elfos que hacen los regalos siguen trabajando.

Los elfos suelen tener problemas y necesitan consultar a Papá Noel para resolverlos. Pero como Papá Noel está muy cansado les dijo a los elfos que lo despierten únicamente si 3 de ellos tenían problemas y que no atendería a más de 3 por vez. Además si algún elfo necesitara ayuda mientras Papá Noel esté ayudando a otros elfos, entonces el primero esperará a que regresen los demás.

Además Papá Noel instruyó a los renos para que lo despierten ni bien el último de los 9 regrese de sus vacaciones para que él pueda preparar el trineo y enganchar a los renos.

Además Papá Noel decidió que si cuando se despierta ve que hay tres elfos esperando por ayuda y además que todos los renos han regresado de sus vacaciones, entonces los elfos tendrán que esperar hasta después de Navidad (asumimos que los renos no vuelven de sus vacaciones hasta inmediatamente antes de la Navidad).

Papá Noel - Solución

```
PAPA NOEL:
despertarPapaNoel.wait()
mutex.wait()
 if(renos == 9)
   renos = 0
   prepararTrineo()
   engancharReno.signal(9)
   repartirRegalos()
 else
   ayudarElfo.signal(3)
mutex.signal()
```

```
RENO:
mutex.wait()
 renos++
 if(renos == 9)
   despertarPapaNoel.signal()
mutex.signal()

engancharReno.wait()
serEnganchadoAlTrineo()
```

```
ELFO:
meAyudaAMi.wait()
mutex.wait()
 elfos++
 if(elfos == 3)
   despertarPapaNoel.signal()
 else
   meAyudaAMi.signal()
mutex.signal()

ayudarElfo.wait()
recibirAyuda()

mutex.wait()
 elfos--
 if(elfos == 0)
   meAyudaAMi.signal()
mutex.signal()
```

Papá Noel - Solución alternativa

```
PAPA NOEL:
despertarPapaNoel.wait()
mutex.wait()
 if(renos == 9)
   renos = 0
   mutex.signal()

   prepararTrineo()
   engancharReno.signal(9)
   repartirRegalos()

 else
   mutex.signal()

   ayudarElfo.signal(3)
   finAyuda.wait(3)

   mutex.wait()
    if (elfos >= 3)
      elfos -= 3
      despertarPapaNoel.signal()
   mutex.signal()
```

```
RENO:
mutex.wait()
 renos++
 if(renos == 9)
   despertarPapaNoel.signal()
mutex.signal()

engancharReno.wait()
serEnganchadoAlTrineo()
```

```
ELFO:
mutex.wait()
 elfos++
 if(elfos == 3)
   despertarPapaNoel.signal()
mutex.signal()

ayudarElfo.wait()
recibirAyuda()
finAyuda.signal()
```

Preguntas?