PROGRAMACIÓN DE SISTEMAS CONCURRENTES Y DISTRIBUIDOS

Dorian Boleslaw Wozniak 817570
Práctica 3 Jaime Velasco Gimeno 816818

DISEÑO:

Se ha realizado el diseño del programa mediante monitores utilizando la técnica de paso de testigo:

Hay N_USER procesos cliente que realizan la siguiente rutina N_TIMES_USER veces:

Esperarán si se va a limpiar el locutorio o ya estan ocupadas las N_CAB cabinas disponibles. Si no se está limpiando el locutorio y hay una cabina libre, entran y ocupan la primera cabina disponible que encuentren. Utilizarán la cabina durante un tiempo aleatorio y saldrán del locutorio y espera un tiempo aleatorio, repitiendo el proceso.

Por otro lado, hay un proceso limpieza con el siguiente algoritmo que se repite N_TIMES_CLEANING veces:

Esperará un tiempo PER_CLEANING; pasado esto avisará que va a limpiar, y espera a que se liberen todas las cabinas. Entonces procederá a limpiar un tiempo aleatorio. Una vez hecho esto, indica que ha acabado de limpiar y repite el proceso.

Hay cuatro secciones críticas: la zona entre que un proceso comprueba si puede entrar al locutorio y ocupa definitivamente una cabina, el momento en el que libera una cabina ocupada, el aviso de que va a limpiar el proceso limpieza, y las órdenes entre que el proceso limpieza espera a que el locutorio se vacíe, limpie las cabinas y avise del fin de la limpieza.

Estas secciones críticas serán gestionadas por tres semáforos: sEnt, que gestionará los procesos cliente bloqueados esperando a poder entrar; sVac, que mantendrá bloqueado el proceso limpieza mientras que no esté el locutorio libre; y sTes, que será el testigo de la implementación y permitirá o no la entrada a las zonas de exclusión mútua

Adicionalmente se requieren los siguientes datos compartidos: los enteros cEnt y cVac, que contarán los procesos cliente y limpieza bloqueados respectivamente; el entero dentro, que cuenta el número de clientes en el locutorio; el booleano limpiar, que será verdad si se va a limpiar el local; y el vector de booleanos cab de tamaño N_CAB, que indica cuales de las cabinas está libre u ocupada.

PSEUDOCÓDIGO:

- CONSTANTES -

```
constant integer N_USER := 20
constant integer N_TIMES_USER := 30
constant integer N_TIMES_CLEANING := 5
constant integer PER_CLEANING := 100
constant integer N_CAB := 4
```

- SEMAFOROS -

```
semaphore sEnt := (0, {cliente, limpieza})
semaphore sVac := (0, {cliente, limpieza})
binary semaphore sTes := (1, {cliente, limpieza})
```

- VARIABLES COMPARTIDAS -

```
integer dentro := 0
integer cEnt := 0
integer cVac := 0
boolean limpieza := false
boolean array cab[1..N_CAB]
```

- VERSIÓN CON AWAITS Y ZONAS DE EXCLUSIÓN MÚTUA -

```
process cliente(i := 1..N_USER):
   for i in 1..N_TIMES_USER
      <await not limpiar and dentro < N_CAB
         dentro := dentro + 1
         integer indice := -1
         for j := 1..n and not indice = -1
            if cab[j] = false
               cab[j] := true
               indice := j
            end
         end
      >

      espera(random)

      <cab[indice] := false
      dentro := dentro - 1>
   end
end

process limpieza:
   for i in 1..N_TIMES_CLEANING
      espera(PER_CLEANING)
      <limpiar = true>
      <await dentro = 0
         espera(random)
         limpiar = false
      >
   end
end
```

- VERSIÓN CON SEMÁFOROS -

```
process cliente(i := 1..N_USER):
   for i in 1..N_TIMES_USER
      wait(sTes)

      if not (not limpiar and dentro < N_CAB)
         cEnt := cEnt + 1
         signal(sTes)
         wait(sEnt)
      end

      dentro := dentro + 1
      integer indice := -1

      for j := 1..n and not indice = -1
         if cab[j] = false
            cab[j] := true
            indice := j
         end
      end

      pasarTestigo()

      espera(random)

      wait(sTes)
      cab[indice] := false
      dentro := dentro - 1
      pasarTestigo()
   end
end

process limpieza:
   for i in 1..N_TIMES_CLEANING
      espera(PER_CLEANING)

      wait(sTes)
      limpiar := true
      pasarTestigo()

      wait(sTes)

      if not dentro = 0
         cVac := cVac + 1
         signal(sTes)
         wait(sVac)
      end

      esperar(random)
      limpiar := false
      pasarTestigo()
   end
end

process pasarTestigo:
   if not limpieza and dentro < N_CAB and cEnt > 0
      cEnt := cEnt - 1
      signal(sEnt)

   else if dentro = 0 and cVac > 0
      cVac := cVac - 1
      signal(sVac)

   else
      signal(sTes)
   end
end
```