Clase Auxiliar 3

Profesor: Luis Mateu
Auxiliares: Juan Manuel Barrios, Hernán Arroyo
29 de Agosto de 2008

1. Equivalencia de mecanismos de sincronización

Implemente una interfaz para semáforos FIFO utilizando monitores. Para su implementación defina métodos análogos a `nMakeSem`, `nWaitSem` y `nSignalSem`, con un mecanismo de asignación de tickets según orden de llegada.

2. Impresora compartida (Examen 1998)

Suponga que en nSystem se dispone de una impresora compartida por todas las tareas. Dos o más tareas pueden usar simultáneamente la impresora, pero en este caso las líneas de la impresora saldrían entremezcladas. En una aplicación se desea evitar el entremezclado de líneas haciendo que cada tarea solicite el acceso exclusivo a la impresora antes de ocuparla y además notifique cuando termina de utilizarla. Por lo tanto una tarea tendrá la siguiente forma:

```
tarea() {
    ...
    obtenerImpresora();
    ...                    /* utilizar impresora */
    devolverImpresora();
    ...
}
```

Además, por razones de ahorro de electricidad, se necesita que la impresora se coloque en modo de bajo consumo cada vez que transcurran 5 minutos sin ser utilizada por ninguna tarea. Para colocar la impresora en modo de bajo consumo invoque el procedimiento `modoBajoConsumo()`. Para volver a usar la impresora cuando está en modo de bajo consumo, invoque el procedimiento `modoUsoNormal()`.

Implemente los procedimientos `obtenerImpresora()`, `devolverImpresora()` y un procedimiento `inicializarImpresora()` que se invoca al inicio de `nMain`. El uso de busy-waiting o consultas periódicas para ver si se han cumplido los 5 minutos es equivalente a no responder la pregunta.

3. Monitores con timeout

Se ha agregado a nSystem el siguiente método para los monitores:

```
void nWaitTimeout(nMonitor mon, int timeout);
```

Su funcionamiento es similar al de `nWait`. Una tarea `T` puede invocar `nWaitTimeout(M, dt)` si y sólo si posee el monitor `M`. En ese caso `nWaitTimeout` libera el monitor `M` y queda a la espera de que una segunda tarea `T'` llame a `nNotifyAll(M)`. La diferencia con `nWait` es que si transcurren `dt` milisegundos desde la invocación de `nWaitTimeout` y ninguna otra tarea ha dado el `nNotifyAll(M)`, `nWaitTimeout` termina su espera (como si se hubiese invocado `nNotifyAll(M)`). Entonces, `nWaitTimeout` solicita nuevamente el monitor `M` y retorna una vez que lo obtiene.

Implemente nuevamente el problema de la impresora compartida pero ahora usando monitores con timeout.