CC41B Sistemas Operativos
Control 2 – Semestre Otoño 2010
Prof.: Luis Mateu

Pregunta 1

Ud. desarrolla un programa que corre en un multiprocesador con 8 cores. En él no existe un núcleo de sistema operativo y la única herramienta de sincronización disponible es el *spin-lock*. En este programa, 4 cores se dedican a producir ítemes que depositan en un *buffer* con el procedimiento `put` y 4 cores extraen los ítemes del buffer con `get` para luego consumirlos. El buffer tiene un tamaño máximo de 20 ítemes. Se le pide a Ud. programar una versión correcta de `put` y `get` usando los spin-locks disponibles. Dado que no hay núcleo de sistema operativo, no existe un scheduler de procesos y no hay herramientas de sincronización avanzadas (como monitores). Por lo tanto cuando por ejemplo Ud. encuentre el buffer vacío en una operación `get`, no tiene otra alternativa que hacer *busy-waiting*.

A modo de recuerdo se incluyen más abajo los procedimientos `put` y `get` para un productor y un consumidor implementados a partir de semáforos de nSystem. Ud. debe reprogramar estos procedimientos, usando solo spin-locks, para el escenario descrito en el párrafo anterior.

```
int ult= 0, prim= 0;
Item buf[20];
nSem vacios /* =nMakeSem(20); */ , llenos /* =nMakeSem(0); */;
```

```
void put(Item it) {
  nWaitSem(vacios);
  buf[ult]= it;
  ult= (ult+1)%20;
  nSignalSem(llenos);
}
```

```
Item get() {
  Item it;
  nWaitSem(llenos);
  it= buf[prim];
  prim= (prim+1)%20;
  nSignalSem(vacios);
  return it;
}
```

Pregunta 2

Se desea agregar *locks* con prioridad a nSystem como herramienta de sincronización nativa. Concretamente la API que Ud. debe implementar es:

- nLock nMakePriLock( ): Entrega un lock con prioridades. En todo instante puede existir a lo más un solo propietario de este lock.
- void nReqLock(nLock lck, int pri): Solicita la propiedad de lck con prioridad pri (un valor entero entre 0 y 3). Si lck está libre, el solicitante adquiere su propiedad de inmediato y nReqLock retorna. Si actualmente otra tarea tiene la propiedad de lck, el solicitante espera hasta adquirir en forma exclusiva lck en una futura llamada de nReleaseLock.
- void nReleaseLock(nLock lck): Libera lck. Si existen varias tareas esperando lck, se entrega su propiedad a la tarea que lo haya solicitado con la prioridad más alta (3 es la más alta, 0 la menor).

Implemente esta API usando los procedimientos de bajo nivel de nSystem (START_CRITICAL, Resume, PutTask, etc.). Ud. *no puede usar* otros mecanismos de sincronización ya disponibles en nSystem, como semáforos, monitores, mensajes, etc.