CC41B Sistemas Operativos

Control 2 – Semestre Primavera 2006

Prof.: Luis Mateu

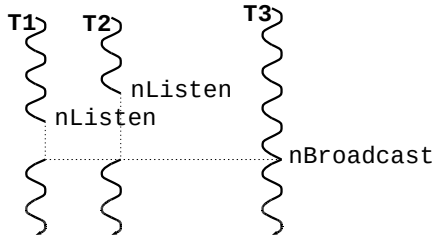
Pregunta 1

Se ha especificado la siguiente API para un sistema de *broadcasts* en nSystem:

Ejemplo de uso	*Descripción*

`nTask emitter= ...;`
`char msg[16];`
`nListen(emitter, msg);`
`... usar msg ...`
	Espera hasta que `emitter` difunda un mensaje. El mensaje difundido se copia en `msg`.

`char msg[16];`
`... llenar msg ...`
`nBroadcast(msg, 16);`
`/* difusión */`
`...`
	Difunde `msg` hacia todas las tareas que estén esperando una difusión de esta tarea. El 2do. parámetro es el largo del mensaje.



Implemente esta API como mecanismo de sincronización nativo de nSystem. Es decir, Ud. debe implementar esta API usando los procedimientos de bajo nivel de nSystem (`START_CRITICAL`, `Resume`, `PutTask`, etc.). Agregue campos al descriptor de tarea nTask, si lo necesita. Ud. *no puede usar* otros mecanismos de sincronización ya disponibles en nSystem, como semáforos, monitores, mensajes, etc.

Pregunta 2

En un núcleo *multi-threaded* varios procesadores pueden trabajar en paralelo dentro del núcleo. El mecanismo de sincronización básico se logra con spin-locks. Se desea agregar un sistemas de monitores para que sea usado internamente en el núcleo. La API es la siguiente:

Operación	*Descripción*
`void kEnter(kMon mon);`	Pide el monitor
`void kExit(kMon mon);`	Libera el monitor
`void kWait(kMon mon);`	Espera hasta un `kNotifyAll`
`void kNotifyAll(kMon mon);`	Activa procesos que esperan en `kWait`, sin perder el monitor

La descripción funcional es idéntica a los monitores de nSystem (o Java). Se especifica que las operaciones kEnter y kExit son para operaciones de corta duración y se implementan en base a spin-locks, y por lo tanto con *busy-waiting*. Cuando una operación requiere esperar por mucho tiempo, los programadores usuarios de esta API deben usar kWait. Se requiere que la implementación de kWait *suspenda el proceso en ejecución y retome un nuevo proceso*, invocando `Resume`. Es decir que kWait no se puede implementar con *busy-waiting* de larga duración.

Se tiene la siguiente implementación incompleta de estos monitores:

```
typedef struct {
  int lck; /* spin lock para kEnter y kExit. */
  ...      /* Agregue lo que se necesita     */
} *kMon;   /* para kWait y kNotifyAll        */
```

```
void kEnter(kMon mon) {
  spinLock(&mon->lck);
}
```

```
void kExit(kMon mon) {
  spinUnlock(&mon->lck);
}
```

Se le pide a Ud. implementar los procedimientos kWait y kNotifyAll. *No modifique* la implementación de kEnter o kExit. Agregue la información adicional que necesite en el descriptor del monitor. Ud. dispone de procedimientos similares a los de nSystem:

`void PutTask(Queue q, Proc p);` `Proc GetTask(Queue q);`	Manejo de colas
`Queue ready_queue;` `int ready_queue_lck;`	Cola de procesos listos para ejecutarse y su *spin-lock*
`void Resume();`	Retoma otro proceso