CC4302 Sistemas Operativos

Control 2 – Semestre Otoño 2013

Prof.: Luis Mateu

Pregunta 1

Se desea implementar un sistema de difusiones en nSystem (*multicast*). Su API es la siguiente:

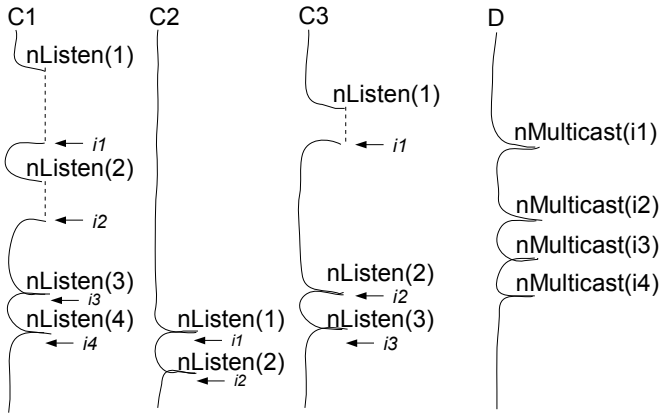
`void nMulticast(void *info);` Difunde una información a todos los interesados. Las difusiones se identifican como 1, 2, 3, etc.
`void *nListen(int id);` Entrega la información asociada a la difusión *id*. Si aún no fue emitida se espera hasta que se emita. Si *id* no es una de las últimas 16 difusiones, se entrega NULL.

El siguiente código muestra un ejemplo de uso de esta API.

```
void difusor() {
  for(;;) {
    Info *info= producir();
    nMulticast(info);
} }
```

```
void consumidor() {
  int id= 1;
  for (;;) {
    Info *info=
        (Info*)nListen(id);
    consumir(info);
    id++;
} }
```

El diagrama de threads de más abajo es un ejemplo de ejecución. El thread D invoca *difusor* y C1, C2 y C3 ejecutan *consumidor*. A diferencia del clásico productor/consumidor en donde cada consumidor recibe un ítem distinto, en este caso *todos los consumidores reciben los mismos ítemes*.



Cuando C1 pide la primera difusión, debe esperar hasta que D emita *i1*. Entonces se entrega *i1*. También debe esperar cuando pide la 2da difusión. Pero por la 3era o 4ta difusión no espera porque ya fueron emitidas. Observe que el thread C2 recién invoca nListen(1) cuando ya se difundió el 4to multicast, y aún recibe la primera difusión correctamente.

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. Ud. sí puede introducir modificaciones en el descriptor de tarea (nTask), agregar estados, etc.

Su implementación debe ser tal que solo guarde las últimas 16 difusiones. Si un thread pide una difusión previa a las últimas 16 difusiones, recibe NULL. Además puede suponer que si la identificación de la última difusión es *k*, entonces un thread puede pedir la difusión *k+1*, pero no una posterior (como *k+2*).

Pregunta 2

Se tiene una máquina *quad-core* en la que no existe un núcleo de sistema operativo. Un core invoca la función *difusor* de la pregunta 1 y los otros 3 cores invocan *consumidor*. Haga una nueva implementación de la API de la pregunta 1 de modo que la sincronización entre los cores se haga mediante *spin-locks*.

Hint: Para hacer que *nListen* espere, declare un spin-lock como una variable local de *nListen*. El spin-lock parte inicialmente cerrado. Coloque un puntero al spin-lock en alguna estructura de datos compartida con el difusor y luego espere con *spinLock*, hasta que *nMulticast* abra el spin-lock con *spinUnlock*.

Recuerde que dado que no hay núcleo de sistema operativo:

- no existe un scheduler de procesos: no puede llamar a `Resume`.
- no hay herramientas de sincronización avanzadas como monitores, semáforos o mensajes.
- si lo necesita, la función `int coreId()` entrega la identificación (entre 0 y 3) del core que invoca *coreId*.

Aún así suponga que Ud. dispone de una implementación de colas FIFO (*FifoQueue*). ¡No olvide evitar *data-races* o *dead-locks*!