CC4302 Sistemas Operativos – Control 2 – Semestre Primavera 2017 – Prof.: Luis Mateu

Pregunta 1

Parte a.- (4 puntos) En una calle hay 10 estacionamientos contiguos al lado derecho y ninguno al lado izquierdo. Se identifican como 0, 1, 2, 3, …, etc. Las tareas de nSystem representan a los automovilistas que solicitan estacionar sus vehículos llamando a la función *nReservar*. Esta función recibe como parámetro el número de estacionamientos contiguos requeridos (*k*). Un automóvil requiere solo 1 estacionamiento, una camioneta requiere 2 que estén contiguos, un camión 3 contiguos pero si tiene remolque podría requerir 5 estacionamientos. Si hay *k* estacionamientos contiguos disponibles, esta función los reserva y retorna de inmediato la identificación del primer estacionamiento en la serie otorgada. De lo contrario, *nReservar* espera hasta que hayan *k* estacionamientos contiguos disponibles. Cuando un automovilista se va invoca la función *nLiberar* indicando el primer estacionamiento que ocupaba (*e*). Esta función libera todos los estacionamientos de la serie otorgada previamente a ese automovilista con *nReservar*, y por lo tanto pueden ser otorgados a otros automovilistas.

Programe las funciones *nReservar* y *nLiberar* como funciones primitivas de nSystem. Esto significa que Ud. debe usar las funciones de bajo nivel de nSystem (*START_CRITICAL*, *Resume*, *PutTask*, etc.). Declare las variables globales que necesite e indique qué nuevos campos requiere agregar al descriptor de tarea. Los encabezados son los siguientes:

```
int nReservar(int k);
void nLiberar(int e);
```

Restricciones:

- No puede usar otros mecanismos de sincronización ya disponibles en nSystem como semáforos, monitores, etc.
- Debe maximizar el uso de los estacionamientos, sin importar que esto se traduzca en hambruna. Por lo tanto no se requiere un orden específico de atención.

Parte b.- (1 punto) Explique cuál es la principal razón por la que un núcleo clásico de un sistema operativo no funciona en máquinas multi-core. ¿Cómo haría que un núcleo clásico sí funcione en máquinas multi-core haciendo un mínimo de modificaciones?

Parte c.- (1 punto) Todos los procesadores tienen una instrucción que desactiva las interrupciones. Explique si sería posible que un programador use esa instrucción para suprimir el mecanismo de *time-slicing* y así impedir que el sistema operativo le quite la CPU a su proceso.

Pregunta 2

I. (4 puntos) Considere una máquina con 8 cores físicos que comparten la memoria, sin un núcleo de sistema operativo y por lo tanto no hay scheduler de procesos. Programe las funciones:

```
int reservar(int k);
void liberar(int e);
```

De manera similar a la pregunta 1, estas funciones permiten que los cores reserven y liberen estacionamientos. Declare las variables globales que necesite señalando su valor inicial. Ud. dispone de la función *coreId( )* que entrega el número del core que la invoca (entre 0 y 7). Use spin-locks para sincronizar los distintos cores. Debe maximizar el uso de los estacionamientos.

Restricciones: No hay *malloc*, *fifoqueues*, *LLMutex*, cola de procesos "ready", etc. Para esperar no le queda otra que hacerlo mediante un spin-lock. No puede recurrir a otra forma de *busy-waiting*.

II. (1 punto) A continuación se establecen 3 objetivos primarios para el desarrollo de un scheduler de procesos. Para cada uno de ellos elija, entre las estrategias de scheduling vistas en clases, la estrategia que mejor cumple ese objetivo:

- Minimizar el número de cambios de contexto implícitos.
- Minimizar el tiempo promedio que transcurre desde que un proceso pasa al estado listo para ejecutarse hasta que pasa nuevamente a un estado de espera.
- Entregar la sensación de avance continuo a los usuarios interactivos.

III. (1 punto) Al programar el núcleo de un sistema operativo, Ud. necesita garantizar la exclusión mutua para incrementar una variable entera global. Confeccione una tabla de 3 columnas por 3 filas, considerando para las columnas las siguientes 3 herramientas: 1 semáforo, 1 spin-lock, inhibición de interrupciones, y para las filas estos 3 tipos de núcleo: clásico mono-core, multi-threaded para mono-core y multi-threaded para multi-core. En cada celda de la tabla indique con un sí o un no la factibilidad de usar la herramienta correspondiente a esa columna para garantizar la exclusión mutua considerando el tipo de núcleo de esa fila. En caso de duda complemente con una explicación.