DSO Febrero 2006

Ejercicio 1 – Procesos y memoria

El sistema operativo es un programa con un alto grado de concurrencia y de asincronía. En cada momento se pueden producir múltiples eventos de forma asíncrona. De esta forma, a lo largo del diseño del sistema operativo debemos ser capaces de sincronizar el acceso a estructuras de datos del núcleo. Son muchos los diferentes escenarios que se pueden producir.

a) Primer escenario: Protección de una estructura de datos del núcleo accedida desde varias llamadas al sistema (llamadas al sistema concurrentes).**[3,5/10 puntos]**
Responda razonadamente a las siguientes preguntas:
¿Cómo se realizaría la sincronización en los siguientes casos?
a1) Suponiendo que se trata de un sistema con un *núcleo no expulsivo*.
a2) Suponiendo que se trata de un sistema con un *núcleo expulsivo*. En este caso, ¿realizaría alguna consideración en relación a las características específicas de la región crítica que se desea proteger?
a3) Suponiendo que se trata de un sistema multiprocesador. En este caso, ¿realizaría alguna consideración en relación a las características específicas de la región crítica que se desea proteger?

b) Segundo escenario: Protección de una estructura de datos accedida desde interrupciones. Suponga una estructura de datos a la que se accede desde varios manejadores de interrupción. En este caso no es necesario diferenciar entre núcleo expulsivo y no expulsivo. **[3/10 puntos]**
Responda razonadamente a las siguientes preguntas.
En un sistema monoprocesador
b1) ¿Se podrían evitar las condiciones de carrera con el uso de semáforos?
b2) ¿Se podrían evitar las condiciones de carrera con el uso de spinlocks?
b3) ¿Qué mecanismo se podría utilizar?

c) Tercer escenario: Protección de una estructura de datos accedida por una llamada al sistema y una interrupción. **[3,5/10 puntos]**
Responda razonadamente a las siguientes preguntas:
¿Cómo se realizaría la sincronización en los siguientes casos?
c1) Suponiendo que se trata de un sistema con un *núcleo no expulsivo*.
c2) Suponiendo que se trata de un sistema con un *núcleo expulsivo*.
c3) En un sistema multiprocesador ambas rutinas deben usar un spinlock y además la rutina menos prioritaria debe inhibir localmente las interrupciones. ¿Por qué hay que inhibir las interrupciones?

Solución

a.- Sincronización entre llamadas al sistema
a1.- En núcleo no expulsivo
En principio, no existe este problema ya que no se permite la ejecución concurrente de llamadas. De esta forma, una llamada al sistema continúa hasta que termina o causa el bloqueo del proceso. Por otra parte, si el proceso se queda bloqueado, cuando continúe su ejecución deberá comprobar que el estado es el correcto (por ejemplo, comprobar una condición que se cumplía antes del bloqueo, ya que es posible que ya no se cumpla).

a2.- En núcleo expulsivo
La solución habitual es implementar un semáforo o algún mecanismo de sincronización equivalente. Estas primitivas permiten que el proceso duerma hasta que el recurso esté disponible. Además, los semáforos funcionan en sistemas monoprocesador y multiprocesador. La expulsión del proceso tampoco crea problemas. Si se expulsa un a proceso que posee un semáforo, un nuevo proceso podría intentar obtener el semáforo. Cuando esto ocurre, el nuevo proceso pasará a estado de bloqueado, hasta que el proceso anterior libere el semáforo.
La otra opción es elevar el nivel de interrupción de forma de manera que se inhiban las interrupciones software asociadas al cambio de contexto involuntario durante el fragmento conflictivo.
En cuanto a la decisión de diseño, si la sección crítica es muy corta, será más adecuado utilizar la estrategia de inhibir las interrupciones software. Hay que tener en cuenta que la solución basada en semáforos incluye una cierta sobrecarga debido a las operaciones vinculadas con los mismos (para lograr la atomicidad de estas operaciones se debe recurrir al mecanismo de elevar el nivel de interrupción). Por otra parte, para secciones críticas de mayor tamaño, será más adecuado el uso de semáforos.

a3.- En multiprocesador
En sistemas multiprocesadores la técnica básica es el spinlock, que se basa en el uso de una variable como un cerrojo. La variable se consulta en un bucle con espera activa hasta que tenga un valor que indique el cerrojo está abierto. Los spinlocks permiten también construir mecanismos como los semáforos.
La sincronización entre llamadas concurrentes se puede conseguir mediante el uso de spinlocks para el control de acceso a las estructuras de datos del sistema operativo (en secciones críticas cortas) o mediante el uso de semáforos (en caso de secciones críticas largas o en las que se puede

producir un bloqueo).

b.- Sincronización entre interrupciones.
Las condiciones de carrera se deben evitar deshabilitando las interrupciones en todas las secciones críticas del manejador de la interrupción (b3). No se puede tomar ninguna otra acción correctora, ya que cualquier otra primitiva de sincronización no servirá. Un semáforo podría bloquear el proceso, por lo que no se puede usar en una rutina de interrupción (b1). Por otra parte, un spinlock puede congelar al sistema: si se interrumpe a un manejador que está accediendo a una estructura de datos, no se puede liberar el spinlock; de esta forma, el nuevo manejador de interrupción se quedará esperando a que se libere el spinlock (b2).

c.- Sincronización entre llamadas al sistema e interrupciones.

c1 y c2.- Tanto en sistemas con núcleo expulsivo como en sistemas con núcleo no expulsivo la prevención de las condiciones de carrera es bastante sencillo. Si se accede a las estructuras de datos del núcleo con las interrupciones locales deshabilitadas, el núcleo no podrá ser interrumpido durante el acceso. Además sabemos que una interrupción nunca podrá ser interrumpida por una llamada al sistema, por lo que no nos tendremos que preocupar por este tipo de situaciones.

c3.- Es necesario inhibir localmente las interrupciones ya que se podría producir un bloqueo en el sistema. Es similar al caso analizado en b2. Si se interrumpe a un manejador que está accediendo a una estructura de datos protegida por un spinlock, el nuevo manejador de interrupción podría quedarse bloqueado esperando a ese spinlock que nunca podrá ser liberado. La forma de evitarlo es inhibir localmente las interrupciones de forma que esta situación no podrá darse nunca.