Examen 2º parcial de Sistemas Operativos Avanzados (6/6/2014)

1) *Explique cómo se resuelven en un sistema multiprocesador los problemas de sincronización que pueden existir entre una llamada al sistema y una rutina de interrupción especificando qué acciones de sincronización se incluyen en el código de llamada y cuáles en la rutina de interrupción.*

En un sistema monoprocesador, para evitar los problemas de sincronización que pueden producirse entre una rutina de interrupción y una llamada al sistema, basta con inhabilitar la interrupción correspondiente durante el fragmento de la llamada al sistema que puede verse perturbado por el tratamiento de la interrupción. Nótese que para este tipo de sistemas no es necesario incluir ningún tipo de código de sincronización en la rutina de interrupción: sólo se requiere en la llamada.

En un sistema multiprocesador, sin embargo, inhabilitar la interrupción no es suficiente ya que la rutina de interrupción puede ejecutarse en otro procesador, en paralelo con la llamada. Es necesario, por tanto, usar un *spinlock* tanto en la llamada al sistema como en la rutina de interrupción para proteger las respectivas secciones críticas. Obsérvese que no sería válida una sincronización basada en semáforos o mutex puesto que estos mecanismos pueden causar bloqueos y en una rutina de interrupción no puede haberlos. Recapitulando, en la rutina de interrupción hay que incluir la solicitud del *spinlock* y la liberación del mismo protegiendo el fragmento de código conflictivo, mientras que en el fragmento de código de la llamada al sistema que hay que proteger, se debe inhabilitar la interrupción y solicitar el *spinlock* a la entrada de ese fragmento y liberar el *spinlock* y rehabilitar las interrupciones a la salida del mismo. Nótese que no es suficiente con usar el *spinlock* sino que también hay que inhabilitar la interrupción puesto que, de no hacerlo, se podría producir un interbloqueo: la llamada al sistema obtiene el *spinlock* y se ve interrumpida localmente por la interrupción conflictiva pero la rutina de interrupción no puede progresar ya que debe conseguir el *spinlock* que está en posesión de la llamada interrumpida.

2) *Responda a las siguientes cuestiones sobre las técnicas de prevención de interbloqueos:* ***(a)*** *¿en qué consisten?* ***(b)*** *¿cuál es su principal inconveniente?* ***(c)*** *describa un esquema basado en esta técnica;* ***(d)*** *¿para qué tipo de recurso son más adecuadas: recursos usados por las aplicaciones o recursos internos del sistema operativo?*

(a) Las técnicas de prevención consiguen que no se produzca un interbloqueo realizando una gestión de recursos que asegure que no se cumpla al menos una de las cuatro condiciones necesarias requeridas por el mismo.

(b) Estas técnicas establecen una serie de restricciones sobre la solicitud de recursos que obligan a que los procesos tengan que solicitar algunos recursos antes de que realmente los necesiten, causando, por tanto, una infrautilización de los mismos.

(c) Teniendo en cuenta que una de las condiciones necesarias es que exista una lista de espera circular entre los procesos involucrados, una estrategia de prevención consiste en ordenar los recursos y obligar a los procesos a que soliciten en ese orden los recursos, aunque no coincida con el orden natural de uso por parte de la aplicación.

(d) Dado que el principal inconveniente de esta técnica es que puede conllevar que un recurso quede reservado pero sin usar durante el intervalo de tiempo que transcurre desde que el recurso ha tenido que ser solicitado debido a las restricciones exigidas por esta técnica hasta que realmente necesita ser utilizado, será más tolerable su aplicación para la gestión de los recursos internos del sistema operativo puesto que el tiempo de uso de los mismos está más acotado a priori (normalmente, el correspondiente a la ejecución de la rutina de tratamiento del evento correspondiente).

3) *Desarrolle los siguientes aspectos sobre la cache de bloques:* ***(a)*** *Fundamento;* ***(b)*** *Políticas de remplazo;* ***(c)*** *Políticas de escritura.*

(a) La enorme diferencia en el tiempo de acceso entre la memoria y los discos conlleva la necesidad del uso de una cache en memoria que almacene parte de la información del sistema de ficheros para lograr de esta forma una solución más eficiente. Como ocurre en otros niveles de la jerarquía de memoria (como, por ejemplo, en la memoria cache del procesador), el éxito de esta técnica se fundamenta en el concepto de proximidad que establece que, después de acceder a una cierta información, es altamente probable que se vuelva a acceder a corto plazo, lo que permite que, aunque el tamaño de la cache de bloques sea mucho menor que el de los discos, sea probable encontrar en la cache la información solicitada por los procesos.

(b) Para el reemplazo de la cache de bloques, se usan las mismas políticas que en el sistema de memoria virtual. Tradicionalmente, la política habitual es LRU, en la que, cuando la cache está llena, se expulsa el bloque que lleva más tiempo sin ser accedido. Nótese que, a diferencia de lo que ocurre con la memoria virtual, en este caso es factible implementar esta política de forma estricta puesto que el sistema operativo gestiona los accesos a la cache (en el caso de la memoria virtual, los accesos a las páginas no pasan por el sistema operativo). Dadas las limitaciones que presenta el algoritmo LRU en aquellos escenarios donde un fichero se accede sólo una vez, existen políticas alternativas, como ARC, para afrontar adecuadamente este tipo de escenarios.

(c) Hay dos alternativas básicas: escritura inmediata, en la que toda escritura, además de copiar la información en la caché de bloques, la escribe en el dispositivo, y escritura diferida, donde los datos sólo se copian en la cache, no siendo escritos en el disco hasta que son expulsados de la cache por la política de reemplazo. En sistemas de propósito general, dada la diferencia en tiempo de acceso entre la memoria y el disco se opta por la segunda alternativa. Sobre el esquema de escritura diferida hay dos posibles extensiones: *delayed-write*, en la que, para acotar el tiempo que pueden tardar en escribirse en el disco los bloques modificados (mientras los bloques modificados no se escriban al disco, se corre el riesgo de perder esa información si se cae la máquina), periódicamente se escriben estos al disco (en UNIX, típicamente cada 30 segundos); *write-on-close*, donde también se escriben a disco los bloques modificados cuando se cierra el fichero.

4) *¿En qué consiste un descifrado genérico como estrategia de detección de virus?*

Es una técnica usada para detectar virus, especialmente ideada para el caso de virus polimórficos, que son aquéllos que para evitar ser detectados por su *firma* (fragmentos de su código), mutan en cada copia (por ejemplo, cifrando parte de su contenido con una clave aleatoria y descifrándose a sí mismo en tiempo de ejecución).

Para superar el reto requerido para la detección de este tipo de virus mutantes, la técnica del descifrado genérico usa un emulador de UCP que ejecuta el código sospechoso sin que éste pueda causar ningún problema a la máquina real, con la intención de que si este ejecutable contiene un virus, éste se descifre a sí mismo quedando visible su código. En paralelo, se ejecuta un escáner que va buscando las posibles firmas de virus conocidos sobre el código emulado según éste evoluciona.