Examen parcial de Sistemas Operativos Avanzados (4/5/2016)

1) *En una comparativa entre sistemas monolíticos y soluciones basadas en micronúcleos, explique de forma razonada pero breve* ***(1)*** *a qué tipo de sistema le afectan menos las desventajas asociadas al uso de un núcleo no expulsivo y* ***(2)*** *en qué tipo de sistema es más factible usar un modelo de interrupciones en vez de uno de procesos.*

(1) La principal desventaja de un núcleo no expulsivo es que proporciona un peor tiempo de respuesta para los procesos, puesto que si se desbloquea un proceso más prioritario que el que está actualmente en ejecución y éste acaba de comenzar el tratamiento de un evento síncrono (como, por ejemplo, una llamada al sistema), la activación del proceso más prioritario se retrasará hasta que se complete el tratamiento. La repercusión de este problema dependerá de cuánto dura el tratamiento de los eventos síncronos en cada sistema: cuanto más largas sean las llamadas al sistema, más se verá afectado el tiempo de respuesta de los procesos.

En un sistema basado en un micronúcleo, el núcleo proporciona un número muy limitado de llamadas al sistema que ofrecen una funcionalidad muy básica y que, por tanto, son muy breves. En consecuencia, este tipo de sistemas se ven mucho menos afectados por el uso de un núcleo no expulsivo.

(2) En el modelo basado en interrupciones, a diferencia del basado en procesos, el tratamiento de todos los eventos, sean síncronos o asíncronos, se completa siempre sin un cambio de proceso en la mitad del mismo (es decir, el tratamiento ejecuta en una sola fase). Esta técnica permite que todos los procesos puedan usar una misma pila de sistema para su ejecución (realmente, una por cada procesador), pero dificulta la programación de las llamadas al sistema e impide implementar mecanismos más sofisticados, como conseguir que un núcleo sea expulsivo o permitir que se produzcan fallos de página en modo sistema, puesto que en ambos casos es necesario que cada proceso tenga su propia pila de sistema.

Las llamadas al sistema en un sistema monolítico son complejas y, de forma natural, requieren varias fases, lo que dificulta su implementación en un sistema con un modelo de interrupciones necesitando usar técnicas algo artificiosas como las continuaciones.

En un sistema basado en un micronúcleo, las llamadas son mucho más sencillas, requiriendo frecuentemente sólo una fase, por lo que es más factible la implementación de un modelo de interrupciones en este tipo de sistemas.

2) *Explique por qué motivo se usa la técnica de co-scheduling (o gang scheduling) en la planificación de máquinas virtuales paralelas y describa en qué consiste el problema de fragmentación asociado a este esquema.*

En un multiprocesador real, los procesadores, obviamente, ejecutan simultáneamente. En un multiprocesador virtual, sin embargo, en principio, no tienen por qué ser así. El planificador del *hipervisor*, en un momento dado, puede asignar un procesador virtual del multiprocesador virtual a un procesador físico y dejar a otro procesador virtual del mismo multiprocesador virtual en el estado de listo para ejecutar sin asignarle ningún procesador físico.

En principio, esa situación no parece problemática, pero en la construcción de un sistema operativo se hacen una serie de suposiciones sobre cómo se comporta la máquina subyacente, como, por ejemplo, que los procesadores siempre están ejecutando, lo que es cierto para un procesador real pero no para uno virtual. Para su sincronización interna, el sistema operativo usa mecanismos de sincronización con espera activa (como, por ejemplo, los *spinlocks*), tal que un procesador que está esperando para entrar en una sección crítica realiza espera activa hasta que el proceso que está actualmente en la sección crítica la completa rápidamente y le deja entrar.

En una máquina real esa espera es muy corta puesto que el procesador en posesión de la sección crítica la completa rápidamente. Sin embargo, en una máquina virtual, el procesador virtual que está en posesión de la sección crítica puede no tener asignado un procesador físico, mientras que el que está en espera activa sí puede tenerlo asignado, con lo que tendrá que esperar un tiempo indefinido, desperdiciando el uso de ese procesador físico mientras tanto.

Una solución habitual a este problema es usar la técnica de *co-scheduling* que plantea asignar simultáneamente un procesador físico a cada procesador virtual de un multiprocesador virtual, asegurando de esta forma que todos los procesadores virtuales de un multiprocesador virtual ejecutan simultáneamente, como en un sistema real.

Esta asignación simultánea puede causar problemas de fragmentación en la asignación de los procesadores físicos. Por ejemplo, en un sistema con 8 procesadores físicos donde hay dos multiprocesadores virtuales con 5 procesadores virtuales cada uno, no se podrían asignar simultáneamente ambos multiprocesadores virtuales, quedándose 3 procesadores físicos sin usar.

3) *Especifique qué dos operaciones no pueden realizarse en el contexto de una interrupción en un sistema operativo de propósito general basado en el modelo de procesos explicando el motivo de esta restricción. Ilústrelo identificando cuál de las dos siguientes versiones, que representan el esqueleto simplificado del manejador de un dispositivo de entrada que opera en modo carácter y usa interrupciones, es errónea remarcando la sentencia incorrecta incluida en este seudo-código.*

Versión 1

```
char *dir_buf; tipo_cola_procesos cola_disp;
int lectura(char *dir, int tam) {
     dir_buf = dir;
     while (tam--) {
          out(R_CONTROL, LECTURA);
          Bloquear(&cola_disp);
}    }
void interrupcion() {
     *(dir_buf++) = in(R_DATOS); // ERROR
     Desbloquear(&cola_disp); }
```

Versión 2

```
char buf; tipo_cola_procesos cola_disp;
int lectura(char *dir, int tam) {
     while (tam--) {
          out(R_CONTROL, LECTURA);
          Bloquear(&cola_disp);
          *(dir++) = buf;
}    }
void interrupcion() {
     buf = in(R_DATOS);
     Desbloquear(&cola_disp); }
```

Desde una rutina de tratamiento de interrupción no se puede acceder al mapa de memoria del usuario, puesto que, dado su carácter asíncrono, no se puede predecir qué proceso está en ejecutando en ese instante. Asimismo, una segunda restricción es que no se puede invocar, directa o indirectamente, una operación de bloqueo desde el contexto de una interrupción, ya que bloquearía el proceso en ejecución, que no tiene nada que ver con ese evento asíncrono, y dejaría incompleto el tratamiento de dicho evento. Nótese que, por su propia condición, el objeto de una interrupción es precisamente lo contrario: desbloquear procesos que estaban esperando su aparición. Las dos restricciones están relacionadas entre sí, puesto que el acceso al mapa de usuario puede causar un fallo de página que es una operación potencialmente bloqueante.

Analizando el código planteado, en la primera versión, dentro del contexto de la interrupción, en la sentencia resaltada, se está accediendo al mapa de usuario del proceso, lo cual es erróneo.

4) *Sobre el algoritmo LRU,* ***(a)*** *explique por qué motivo es difícil lograr una implementación estricta de este algoritmo para un sistema de gestión de memoria y, sin embargo, es perfectamente factible para la caché de un sistema de ficheros;* ***(b)*** *describa qué limitación tiene este algoritmo que ha causado que se propongan nuevos algoritmos (LRU/K, ARC,...) para superarla.*

(a) Para poder implementar este algoritmo se requiere conocer el orden de los accesos. En un sistema de gestión de memoria, esos accesos se corresponden con los realizados por las instrucciones LOAD/STORE del procesador (cada instrucción de este tipo cambiaría el orden LRU de las páginas), por lo que sólo un componente hardware, como es la MMU, puede ser capaz de mantener esta información que refleja el orden de acceso. Se podría construir un procesador con una MMU que gestionara por hardware esta información, pero sería un diseño, por un lado, complejo y, por otro lado, específico para ese algoritmo, no dejando que el diseñador del sistema operativo pudiera definir libremente su algoritmo de reemplazo. La MMU de los procesadores estándar sólo gestiona normalmente un bit de referencia y uno de modificado.

En el caso de la caché de un sistema de ficheros, los accesos se corresponden con las operaciones de lectura y escritura (las llamadas al sistema correspondientes) que realizan las aplicaciones, con lo que no se requiere ningún soporte hardware específico para implementar en este caso el algoritmo, puesto que es el sistema operativo, y no un componente hardware, el que gestiona estos accesos.

(b) El algoritmo LRU sólo tiene en cuenta cuándo fue el último acceso a cada página, en el caso de un sistema de gestión de memoria, o a cada bloque de un fichero, si se trata de la caché de un sistema de ficheros, no teniendo en cuenta otros aspectos como la frecuencia de acceso al objeto correspondiente. En consecuencia, cuando un proceso sólo tiene que acceder una vez a todas las páginas de una región (o a todos los bloques de un fichero), éstas pasan a ser las más recientemente usadas y no serán expulsadas hasta que alcancen el final de la lista según se vayan accediendo otras páginas, aunque realmente el proceso no las va a volver a usar. Para evitar esta patología, se han ideado algoritmos (LRU/K, ARC,...) que, además de tener en cuenta cuán recientemente se ha accedido a una página, también usan la información del número de accesos que ha habido a esa página.