Examen parcial de Sistemas Operativos Avanzados (6/5/2015)

1) *Describa las diferencias entre un sistema basado en un micronúcleo y uno en un exonúcleo, explicando qué interfaz ofrece cada tipo de sistema a las aplicaciones.*

Ambas soluciones tienen en común el intento de minimizar el tamaño del núcleo, es decir, la parte del sistema operativo que ejecuta en modo privilegiado. Sin embargo, los objetivos de cada sistema son significativamente diferentes.

En un sistema basado en un micronúcleo lo que se busca es construir un sistema operativo más fiable y flexible diseñándolo de manera que una mínima parte de su código ejecuta en modo privilegiado, mientras que la funcionalidad restante se implementa como servidores, que ejecutan en modo usuario, comunicándose entre sí. La interfaz que el sistema operativo ofrece a las aplicaciones es la misma que la de un sistema operativo convencional.

En el caso de una solución basada en un exokernel, se pretende ofrecer a cada aplicación sólo el nivel de abstracción que requiere (p.e. un SGBD puede preferir trabajar con bloques que con ficheros). Para ello, el núcleo del sistema, que ejecuta en modo privilegiado, sólo ofrece las abstracciones básicas y existe una serie de bibliotecas de usuario que implementan distintas abstracciones, de manera que cada aplicación usa aquellas bibliotecas que le ofrecen justamente las abstracciones que requiere usar.

2) *Responda de forma razonada, pero breve (una frase), las siguientes cuestiones sobre núcleos expulsivos y no expulsivos.*

a) *¿Qué tipo de núcleo se vería más perjudicado en lo que se refiere a su agilidad por la incorporación de llamadas al sistema largas y no bloqueantes?*

Un núcleo no expulsivo, puesto que en este tipo de núcleos la expulsión de un proceso que está realizando una llamada al sistema se difiere hasta que se completa, o se bloquea, dicha llamada, que, al ser larga y no bloqueante, afecta negativamente al tiempo de respuesta y, por tanto, a la agilidad del sistema.

b) *¿Para qué tipo de sistema, servidor o desktop, sería, en principio, más necesario un núcleo expulsivo?*

Normalmente, en un sistema de tipo *desktop* se utilizan aplicaciones interactivas que son más sensibles a los tiempos de respuesta que el tipo de software que suele ejecutar en un servidor, por lo que, en principio, resulta más necesario un núcleo expulsivo para ese tipo de sistemas más interactivos al ofrecer un mejor tiempo de respuesta.

c) *¿Qué tipo de núcleo es intrínsecamente más robusto y fiable?*

Al no permitir concurrencia entre llamadas al sistema, un núcleo no expulsivo es intrínsecamente más fiable.

d) *¿Qué tipo de núcleo es más eficiente en cuanto a mayor trabajo útil realizado (menor sobrecarga)?*

El núcleo expulsivo lleva asociada una mayor sobrecarga debido a la necesidad de usar mecanismos de sincronización para controlar las condiciones de carrera que surgen al permitir la concurrencia entre llamadas (nótese también que en un núcleo expulsivo puede producirse un número mayor de cambios de contexto involuntarios como, por ejemplo, en el caso de que durante una llamada al sistema bloqueante de un proceso se desbloquee un proceso más prioritario antes del bloqueo del primer proceso).

e) *¿Qué tipo de núcleo sería más fácil adaptar de un sistema monoprocesador a un multiprocesador?*

En un núcleo expulsivo para un monoprocesador están resueltos todos los problemas de sincronización debidos a la ejecución concurrente, aunque no paralela, de las llamadas al sistema, por lo que está mejor preparado para ser adaptado a un sistema multiprocesador extendiendo los mecanismos de sincronización que ya utiliza al contexto de un multiprocesador.

3) *El planificador de Linux usa dos esquemas jerárquicos para llevar a cabo su labor: la planificación de grupos (group scheduling) y los dominios de planificación (scheduling domains). Explique el objeto de cada esquema ilustrándolo brevemente con el ejemplo de un sistema que realiza una planificación equitativa entre usuarios y que dispone de varios paquetes de cores con hyperthreading.*

La técnica del *group scheduling* permite agrupar jerárquicamente procesos para, de esta forma, poder implementar cualquier esquema de planificación equitativa (*fair-share scheduling*), puesto que, además de entre los procesos, permite repartir equitativamente el tiempo de procesador entre los grupos definidos en el sistema. Así, por ejemplo, si se pretende repartir el tiempo entre usuarios en vez de directamente entre procesos (p.ej. con dos usuarios, si un usuario tiene un único proceso, éste recibirá la mitad del tiempo de procesador, con independencia de cuántos procesos tenga el otro usuario), bastaría con definir un grupo por cada usuario.

El esquema jerárquico de los dominios de planificación permite que el sistema operativo gestione adecuadamente la topología jerárquica presente en un multiprocesador actual (*hyperthreading*, core, paquete, nodos NUMA,...). Cada dominio de planificación comprende un conjunto de procesadores organizados en grupos y se encarga de mantener equilibrada la carga entre esos grupos. Los dominios de planificación se organizan de manera jerárquica reflejando la topología del multiprocesador. En un sistema como el planteado en el enunciado, en el nivel inferior, habrá un dominio de planificación por cada *core* que se encargará de mantener equilibrada la carga entre los *threads* de ese *core* (en este nivel, hay un grupo por cada *thread* del core). En el nivel intermedio, existirá un dominio de planificación por cada paquete, que tratará de equilibrar la carga entre los *cores* de ese paquete (en este nivel, hay un grupo por cada *core* del paquete). En el nivel superior, habrá un único dominio de planificación cuyo objetivo será equilibrar la carga entre los grupos que gestiona, que corresponden a cada paquete del sistema.

4) *Indique de forma razonada, pero breve (una frase), en cuáles de las siguientes operaciones puede ser necesario realizar una invalidación parcial o total de la TLB en un sistema que usa una TLB gestionada por hardware que no incluye identificador de proceso.*

a) *Cambio de contexto entre threads del mismo proceso.*

No hay que invalidar la TLB puesto que los dos *threads* comparten el mismo mapa de memoria.

b) *Cambio de contexto entre threads de distinto proceso.*

Hay que invalidar todas las entradas de la TLB, exceptuando las que correspondan al sistema operativo, ya que se ha cambiado el mapa de memoria y, por tanto, las traducciones incluidas en la TLB han dejado de ser válidas.

c) *Creación de una nueva región (p.e. proyección de un archivo: mmap).*

No hay que realizar ningún tipo de invalidación en la TLB puesto que la creación de una región no afecta a las tablas de páginas asociadas al proceso y, al fin y al cabo, la TLB es una cache de esas tablas de páginas.

d) *Eliminación de una región existente (p.e. desproyección de un archivo: munmap).*

La operación de eliminar una región implica marcar como inválidas todas las entradas de las tablas de páginas del proceso que correspondan a páginas de la región que están residentes, lo que conlleva a su vez invalidar las entradas que pueda incluir la TLB correspondientes a dichas páginas.

e) *Poner a 0 el bit de modificado de una entrada de la tabla de páginas después de escribir la página a memoria secundaria.*

Es necesario invalidar la entrada de la TLB correspondiente a esa página, en el caso de que esté presente, para evitar que el estado del bit de modificado en la entrada de la TLB no corresponda con el que aparece en la entrada de la tabla de páginas (nótese que en caso de no hacerlo, si se realiza una operación de escritura en esa página, la MMU no pondría a 1 el bit de modificado de la entrada de la tabla de páginas).