Examen 1er parcial de Sistemas Operativos Avanzados (6/6/2014)

1) *¿Cuáles son las ventajas y/o inconvenientes del diseño de sistemas operativos basados en microkernels en lo referente a aspectos de rendimiento y seguridad? ¿A lo largo de las diferentes generaciones de microkernels, cuándo se han ido mejorando estos aspectos?*

El grado de seguridad de un sistema operativo está condicionado por las vulnerabilidades que presenta, es decir, por los errores existentes en el código del mismo que pueden ser explotados para comprometer su seguridad. El efecto de estos ataques dependerá de en qué nivel de privilegio ejecuta el código erróneo que presenta esa vulnerabilidad. Como consecuencia de ello, una de las reglas básicas para mejorar la seguridad de un sistema es el principio de mínimo privilegio: un determinado código debería ejecutar sólo con el nivel de privilegio requerido para llevar a cabo su labor.

Aplicando directamente este principio y teniendo en cuenta que el máximo privilegio en un sistema es la ejecución de código con el procesador en modo privilegiado, se concluye que un sistema operativo basado en un micronúcleo tendrá un mejor nivel de seguridad frente a uno monolítico al reducir muy significativamente el volumen de código que ejecuta en modo privilegiado. Esta reducción ha permitido incluso en algunos casos poder verificar formalmente la corrección del código de un micronúcleo.

En cuanto al rendimiento, ese es el punto débil de los sistemas operativos basados en micronúcleos. En un sistema monolitico la solicitud de un servicio al sistema operativo implica únicamente dos cambios de modo: de usuario a sistema al invocar la llamada al sistema y de sistema a usuario al completarse la misma. En un sistema basado en micronúcleo, la funcionalidad del sistema está implementada como una colección de servidores que ejecutan en modo usuario que se comunican entre sí mediante paso de mensajes. Por tanto, la solicitud de un servicio requiere enviar un mensaje (lo que conlleva dos cambios de modo al tener que invocar una llamada del micronúcleo, así como la gestión por parte del mismo de la transferencia del mensaje entre los procesos involucrados) al servidor correspondiente que, después de un cambio contexto, procesará dicho mensaje lo que, en muchas ocasiones, puede conllevar el envío de otro mensaje a un servidor adicional y, finalmente, los sucesivos mensajes de respuesta con los cambios de contexto asociados (por ejemplo, una aplicación solicita en servicio que implica enviar un mensaje al servidor del sistema de ficheros correspondiente que, a su vez, envía un mensaje al servidor que gestiona un determinado tipo de disco).

Ante toda esta sobrecarga, las sucesivas generaciones de micronúcleos han intentado optimizar las operaciones de cambio de contexto (haciendo, por ejemplo, que no sea necesario invalidar la TLB durante las mismas) y las transferencias de mensajes (usando técnicas de gestión de memoria para evitar tener que copiar el contenido de los mensajes entre los espacios de memoria de los procesos involucrados en la transferencia).

2) *Indique las diferencias entre una interrupción software de sistema y una de proceso. Ponga un ejemplo para cada uno de los tipos de interrupción software.*

Se trata de mecanismos que crean por software un evento con un comportamiento similar a una interrupción real, compartiendo su carácter asíncrono, y que permiten diferir la ejecución de una determinada operación al contexto pertinente para la misma.

En cuanto las interrupciones software de sistema, se integran en el esquema general de interrupciones del sistema con una prioridad mínima (inferior a la de todas las interrupciones hardware) y se invocan desde el contexto de una interrupción para diferir la ejecución de operaciones que conllevan cierta sobrecarga fuera del contexto de dicha interrupción. Téngase en cuenta que si se realiza un procesamiento largo dentro de una rutina de interrupción hardware, el sistema puede tener peor tiempo de respuesta (todas las interrupciones de menor prioridad deberán esperar hasta que se realice ese procesamiento complejo) e incluso podría perderse alguna interrupción. Con las interrupciones software de sistema, la rutina de tratamiento de la interrupción hardware (por ejemplo, la interrupción del teclado), una vez realizado el trabajo urgente y rápido requerido (obtener el código de la tecla pulsada), delega el trabajo más pesado activando una interrupción software de sistema (traducir del código de tecla al carácter correspondiente

mediante la tabla de traducción de teclado instalada teniendo en cuenta las teclas modificadoras y pudiendo conllevar labores de edición como, por ejemplo, borrar la línea tecleada hasta el momento). Como la rutina de tratamiento de la interrupción software tiene una prioridad inferior a las interrupciones hardware, mientras lleva a cabo la operación más pesada, no hará esperar a las interrupciones de los dispositivos.

Con respecto a las interrupciones software de proceso, son similares a las de sistema pero van dirigidas a un determinado proceso. Como ocurre con las interrupciones software de sistema, sólo se tratarán cuando no haya interrupciones hardware pendientes en el sistema, pero, en este caso, además, debe estar en ejecución el proceso al que van dirigidas. Nótese que en el momento en que se active la interrupción software de proceso, el proceso al que va destinada puede estar en ejecución, ya sea en el mismo procesador donde se activa o en uno diferente (en cuyo caso se requiere una IPI para avisar al procesador destino) o puede estar bloqueado o listo para ejecutar. A continuación, se muestran un ejemplosdel uso de este tipo de interrupción para la planificación de procesos en un sistema monoprocesador.

Cuando se detecta dentro del contexto del tratamiento de una rutina de interrupción que se debe expulsar el proceso en ejecución, se activa una interrupción software de proceso dirigida a este proceso para diferir el cambio de contexto involuntario hasta que las condiciones sean las propicias: se haya completado todo el tratamiento de interrupciones en el sistema y, en caso de un núcleo no expulsivo, no esté pendiente de completar el tratamiento de un evento síncrono. Dado que esta interrupción software de proceso para la planificación tiene menos prioridad que el resto de las interrupciones, e incluso menor que los eventos síncronos en el caso de un núcleo no expulsivo, se cumplirán automáticamente las condiciones requeridas.

3) *¿Qué aspecto ignorado por los esquemas de planificación del procesador tradicionales es el punto fuerte de los algoritmos de tipo fair-share scheduling? Ilustre la explicación aplicándola al ejemplo de un sistema con dos usuarios tal que el primer usuario tiene dos procesos activos mientras que el segundo sólo tiene uno.*

Los algoritmos de planificación convencionales reparten equitativamente el procesador entre los *threads*/procesos listos para ejecutar existentes en el sistema, con independencia de a qué aplicación o usuario pertenecen. De esta forma, en el caso planteado en el enunciado, cada uno de los tres procesos obtendrá un tercio del tiempo del procesador, lo que conlleva que el primer usuario recibirá dos tercios de ese recurso mientras que el segundo conseguirá el otro tercio, lo que no es equitativo desde el punto de vista de los usuarios.

Sin embargo, los algoritmos de planificación *fair-share* permiten ser configurados para repartir de forma equitativa el tiempo de procesador entre los diversos grupos de entidades planificables que haya definido el administrador del sistema estableciendo una jerarquía con pesos asignados a las mismas. En el caso planteado en el enunciado, si el administrador pretende que haya equitatividad entre los usuarios, puede definir un primer nivel de planificación con un grupo por cada usuario asignándoles el mismo peso, de manera que se repartan equitativamente el tiempo del procesador. Los procesos de cada grupo se repartirán el tiempo asignado a su usuario. Así, en el ejemplo planteado en el enunciado, si se aplica una política de equitatividad entre usuarios, los dos procesos del primer usuario obtendrán cada uno un cuarto del tiempo del procesador, mientras que el único proceso del segundo usuario recibirá la mitad.

4) *Describa el algoritmo del reloj (o de la segunda oportunidad) especificando qué acciones lleva a cabo a la hora de determinar qué página residente será reemplazada. ¿Qué ventajas presenta este algoritmo con respecto al LRU?*

Este algoritmo organiza las páginas residentes en memoria en orden FIFO formando una lista circular tal que la cabecera de la lista corresponde a la página que lleva más tiempo residente en memoria (la denominación de algoritmo del reloj proviene de que se puede visualizar la lista circular como la esfera de un reloj y el puntero a la cabecera como su aguja). Además, requiere que se gestione un bit de referencia por cada página, lo que es habitual en las MMU de muchos procesadores.

Cuando se produce un fallo de página, se consulta la página en la cabecera de la lista, como también hace el algoritmo FIFO, pero, a diferencia de éste, si esta página tiene activo el bit de referencia, se le da una *segunda oportunidad*, desactivando este bit y avanzando la aguja del reloj a la siguiente página residente (como consecuencia de este avance, la página a la que se le ha dado esta segunda oportunidad pasa a estar al final de la lista circular), Este proceso se repite hasta que se encuentra una página residente con el bit de referencia desactivado, que será la que resultará reemplazada.

Este algoritmo, a diferencia del puramente FIFO, tiene en cuenta si se ha accedido a una página recientemente para tomar sus decisiones, como también lo hace el algoritmo LRU. Sin embargo, a diferencia de este último, no tiene una información detallada de en què orden se han producido los accesos (simplemente, conoce si se ha vuelto a acceder a la página en el tiempo que transcurre durante una rotación completa de la aguja del reloj), lo que puede conllevar peores decisiones a la hora de seleccionar la página a reemplazar. Su ventaja frente al algoritmo LRU es que la información que requiere (la gestión de un bit de referencia) es proporcionada por la MMU de la mayoría de los procesadores, mientras que la información que necesita un algoritmo LRU requeriría construir una MMU específica.