Diseño de Sistemas Operativos. Septiembre de 2011.
Ejercicio de procesos

Se pretende analizar en qué situaciones resulta ventajoso usar un sistema operativo con un núcleo expulsivo y en cuáles no presenta ningún beneficio frente a utilizar uno con un núcleo no expulsivo. Para este análisis, se plantea estudiar el caso de una máquina que usa una planificación basada en prioridades de carácter expulsivo y que está dedicada exclusivamente a ejecutar una aplicación con dos *threads*: el *thread* A, que permanece bloqueado hasta que le llega una petición de servicio (por ejemplo, esperando datos del teclado), procesándola en ese instante y volviéndose a bloquear, y el *thread* B, de menor prioridad, que ejecuta "de fondo" sin bloquearse en ningún momento. Para cada uno de los supuestos planteados en los siguientes apartados, debe analizar si sería beneficioso utilizar un núcleo expulsivo en ese caso. En aquellas situaciones donde no sea ventajoso, explique por qué motivo, mientras que en las que sí resulte beneficioso, plantee una traza de ejecución que lo ilustre especificando para cada tipo de núcleo todas las activaciones del sistema operativo y los cambios de contexto voluntarios e involuntarios que se produzcan.

a) Antes de analizar los distintos casos planteados, explique, de forma genérica, qué ventajas y desventajas ofrece un núcleo expulsivo frente a uno no expulsivo.

b) Realice el análisis del posible beneficio de un núcleo expulsivo en el caso de que el *thread* B ejecutara siempre en modo usuario (es decir, no hace llamadas al sistema ni provoca fallos de página).

c) Repita el análisis del apartado anterior considerando en este caso que el *thread* B intercala su ejecución en modo usuario con la invocación de una cierta llamada al sistema no bloqueante, no provocando fallos de página.

c1) ¿Cómo influiría la mayor o menor duración de esa llamada al sistema en hacer más o menos ventajoso el uso de un núcleo expulsivo?

c2) Suponiendo que esa llamada al sistema usa el mecanismo de inhibición de expulsión para crear una sección crítica dentro de la misma, ¿influiría la mayor o menor duración de dicha sección crítica en hacer más o menos ventajoso el uso de un núcleo expulsivo?

d) Realice el mismo análisis del apartado *b* suponiendo ahora que el *thread* B **no** realiza llamadas al sistema pero sí provoca fallos de página no bloqueantes.

d1) ¿Cómo influiría la mayor o menor duración de la rutina de tratamiento de los fallos de página en hacer más o menos ventajoso el uso de un núcleo expulsivo?

d2) Ponga ejemplos de situaciones donde la rutina de tratamiento de un fallo de página no sea bloqueante.

e) Por último, repita el análisis del apartado *b* considerando que el *thread* B de la aplicación realiza una cierta llamada al sistema no bloqueante, dentro de la cual se puede producir un fallo de página no bloqueante.

e1) Ponga un ejemplo de una situación donde dentro de una llamada al sistema se produzca un fallo de página.

e2) Ponga un ejemplo de una situación donde dentro de una rutina de interrupción se produzca un fallo de página.

f) Analice el posible beneficio de un núcleo expulsivo en una aplicación donde todos los *threads* tuvieran la misma prioridad.

g) Explique cómo influiría la mayor o menor duración de las rutinas de interrupción de los dispositivos en hacer más o menos ventajoso el uso de un núcleo expulsivo, y cómo lo haría la duración de las rutinas de interrupción software de sistema asociadas a los distintos dispositivos.

Solución

a) El principal beneficio de un núcleo expulsivo es que mejora el tiempo de respuesta del sistema al reducir cuánto tiene que esperar un proceso desde que pasa a estar listo para ejecutar hasta que realmente ejecuta. Así, en un núcleo expulsivo, cuando se desbloquea un proceso de mayor prioridad que el que está en ejecución, sólo debe esperar para ponerse a ejecutar a que concluya el tratamiento de los eventos

asíncronos que estén activos (interrupciones de los dispositivos e interrupciones software de sistema), si es que los hubiera. En el caso de un núcleo no expulsivo, sin embargo, se debe esperar también a que se complete el tratamiento de los eventos síncronos que puedan estar activos (llamada al sistema o fallos de página), en caso de que los haya, lo que aumenta significativamente el tiempo de respuesta de los procesos.

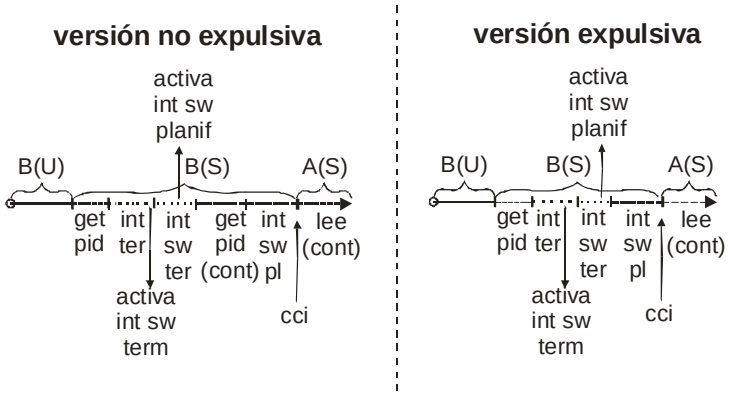
Esta ventaja conlleva un aumento de la concurrencia en el sistema, al permitir que se ejecuten de forma concurrente el tratamiento de eventos síncronos de distintos procesos. Sin embargo, eso implica un tipo de núcleo potencialmente menos fiable, al aumentar significativamente los problemas de sincronización, y menos eficiente globalmente, por la sobrecarga introducida por los mecanismos de sincronización introducidos para eliminar dichos problemas, así como por el aumento significativo del número de cambios de contexto que se producen en el sistema.

b) Para responder a la pregunta planteada, que pretende comparar ambos tipos de núcleos sobre el ejemplo propuesto, hay que analizar el tiempo de respuesta del *thread* de mayor prioridad evaluando para cada tipo de núcleo cuánto tiempo puede transcurrir desde que se desbloquea ese *thread* hasta que prosigue su ejecución expulsando al de menor prioridad.

Dado que el *thread* de menor prioridad nunca ejecuta eventos síncronos, no hay diferencia entre usar un tipo de núcleo u otro: en ambos casos sólo hay que esperar a que se complete el tratamiento de los eventos asíncronos para que se active el *thread* de mayor prioridad realizándose un cambio de contexto involuntario en ese momento.

c) En este caso si puede mejorar el tiempo de respuesta del *thread* de mayor prioridad usando un núcleo expulsivo si el desbloqueo de ese *thread* se produce justo cuando el *thread* de menor prioridad estaba ejecutando una llamada al sistema no bloqueante, puesto que el cambio de contexto involuntario se tendrá que diferir hasta que se complete esa llamada en el caso de un núcleo no expulsivo.

Las siguientes trazas muestran esta diferencia. En ellas se ha supuesto que el *thread* de mayor prioridad está esperando datos del terminal y que el *thread* de menor prioridad está realizando una llamada `getpid` en el momento que se produce la interrupción del terminal, en cuyo tratamiento se ha considerado que se utiliza una interrupción software de sistema para diferir las operaciones menos urgentes de ese tratamiento.



En las trazas se puede observar como en el núcleo no expulsivo el cambio de contexto debe esperar hasta que se complete la llamada, lo que causa un peor tiempo de respuesta.

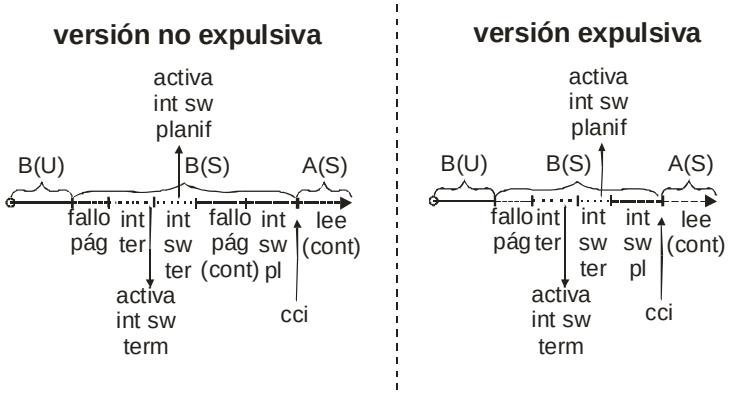
c1) Dado que en un núcleo no expulsivo el cambio de contexto involuntario se tiene que diferir hasta que se complete la llamada en curso, cuanto más larga sea la llamada más beneficioso será usar un núcleo expulsivo. Nótese que, precisamente por esta conclusión, es más importante dotar de un carácter expulsivo al núcleo de un sistema monolítico que al de un sistema basado en un micro-núcleo.

c2) Durante el intervalo de tiempo que se han inhibido las expulsiones para crear una sección crítica entre eventos síncronos, el núcleo se comporta como no expulsivo aunque lo sea. Por tanto, cuanto más largas sean esas secciones críticas menos beneficioso será usar un núcleo expulsivo. Es por ello que este mecanismo sólo se debe usar para secciones críticas muy cortas.

d) Al tratarse también de un evento síncrono, este caso es similar al anterior y nuevamente será beneficioso usar un núcleo expulsivo. Con este tipo de núcleo se mejora el tiempo de respuesta del *thread* de mayor prioridad si el desbloqueo de ese *thread* se produce justo cuando el *thread* de menor prioridad

estaba en el tratamiento de un fallo de página no bloqueante, ya que el cambio de contexto involuntario se tendrá que diferir hasta que se complete dicho tratamiento en el caso de un núcleo no expulsivo.

Las siguientes trazas ilustran ese beneficio. Obsérvese que son muy similares a las del apartado anterior, cambiando simplemente la llamada al sistema por el tratamiento de un fallo de página.



d1) Siguiendo con las similitudes con el apartado previo, en este caso ocurre lo mismo: cuanto más largo sea el tratamiento de un evento síncrono (un fallo de página en este caso) más beneficioso será usar un núcleo expulsivo.

d2) En el ejemplo propuesto en el enunciado se ha planteado que el fallo de página no es bloqueante, es decir, que no provoca ningún acceso al disco. A continuación, se analiza en qué situaciones puede darse ese comportamiento.

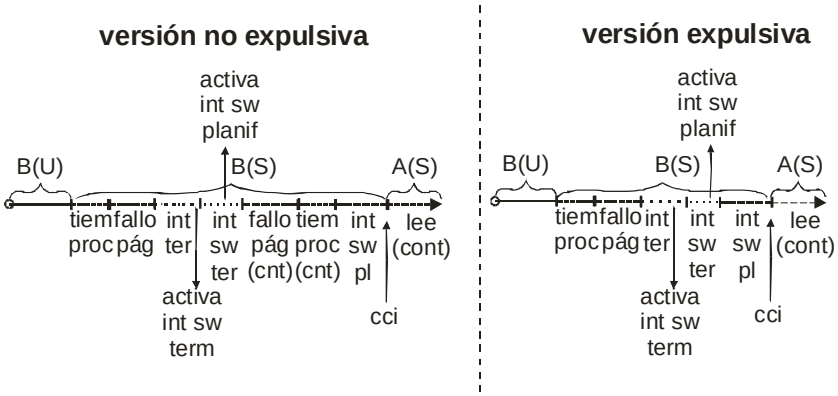
En el peor de los casos en lo que se refiere a accesos al disco, un fallo de página puede causar dos operaciones: la escritura en disco de una página modificada que se va a expulsar para liberar un marco para la nueva, y la lectura de la nueva página del disco.

Para que no se produzca el primer acceso (el de escritura en el disco), basta con que haya marcos libres o, incluso, aunque no los haya, es suficiente con que la página expulsada no esté modificada.

En cuanto a que no se requiera un acceso de lectura a disco, esta situación se producirá en el caso de páginas que pertenezcan a regiones anónimas (sin soporte, como, por ejemplo, la región de datos sin valor inicial, el *heap* o la pila) que todavía no se hayan modificado. Asimismo, también se dará este comportamiento si la página requerida se encuentra en la caché de páginas.

e) Este caso es una combinación de los dos anteriores y, por ello, en él será especialmente beneficioso el uso de un núcleo expulsivo puesto que elimina la necesidad de esperar a que se complete tanto la llamada al sistema como el fallo de página anidado en la misma.

Para las trazas comparativas, vamos a suponer en este caso que, en el momento en el que se produce la interrupción de teclado, el *thread* de menor prioridad está ejecutando una llamada al sistema no bloqueante que ha causado un fallo de página no bloqueante. Por ejemplo, podría tratarse de una llamada que obtiene la contabilidad del tiempo de procesador consumido por el proceso hasta ese instante (en el caso de UNIX, la llamada `times`) a la que se le ha pasado como parámetro una variable sin valor inicial incluida en una página que no se había modificado hasta entonces, habiendo, además, marcos libres en el sistema.



e1) En la propia traza se ha planteado un ejemplo. A continuación, se analiza de una manera más general y se proponen ejemplos adicionales.

Cualquier llamada que reciba como parámetro la dirección de una zona en el mapa de memoria de usuario del proceso y que, obviamente, acceda a la misma ya sea para obtener información o para depositarla, podrá causar fallos de página si las páginas involucradas no está residentes.

Como ejemplos de llamadas que obtienen información de la zona recibida como parámetro estarían todas las llamadas de escritura (`write`), de envío de información (`send`) o que pasan por referencia una cadena de caracteres (`open`) u otra estructura (`utimes`).

En cuanto a llamadas que depositan información en la zona recibida como parámetro estarían todas las llamadas de lectura (`read`), de recepción de información (`recv`) o que pasan por referencia una estructura (`stat`).

e2) Recuerde que bajo ninguna circunstancia puede ocurrir un fallo de página dentro de la rutina de tratamiento de una interrupción, ya que al tratarse de un evento asíncrono no se conoce a priori en el contexto de qué proceso ejecutará y, por tanto, no debe acceder al mapa de usuario del proceso dentro del tratamiento del mismo.

f) El único objetivo directo de un núcleo expulsivo es mejorar el tiempo de respuesta. Por otro lado, hay que tener en cuenta que el tiempo de respuesta es un parámetro que cobra sentido en un sistema con prioridades, donde es crucial (incluso crítico en el caso de un sistema de tiempo real; pero incluso en mi máquina de uso personal no me agrada observar discontinuidades en la reproducción de un vídeo) que un proceso más importante desaloje lo antes posible a uno que lo es menos.

Por todo ello, en un sistema sin prioridades entre los *threads*, el efecto beneficioso de usar un núcleo expulsivo es totalmente discutible. Suponiendo que se usa *round-robin* para repartir el tiempo entre los *threads*, ¿para qué nos sirve que cuando se le acabe la rodaja a un *thread* le expulsemos inmediatamente aunque esté en medio del tratamiento de un evento síncrono? Si nos fijamos en el tiempo de respuesta, su beneficio es prácticamente nulo, puesto que dicho parámetro vendrá condicionado básicamente por cuántos *threads* están delante en la cola de listos cuando se desbloquea el *thread* de nuestro interés. Por lo que se refiere al reparto equitativo del tiempo del procesador, tampoco nos aporta gran cosa. En principio, podría parecer más equitativo el núcleo expulsivo, puesto que el que no lo es le “regala” un tiempo extra a la rodaja para que se complete el tratamiento del evento síncrono pendiente. Sin embargo, ese tiempo es despreciable con respecto al tamaño de la rodaja, que, además, ya de por sí no es de todo preciso (por ejemplo, la interrupción de reloj que marca el fin de rodaja puede llegar en un momento que las interrupciones están inhibidas). Además, el concepto de repartir el tiempo de procesador en rodajas es sólo relativo en el sentido de que durante su rodaja el proceso tendrá que “acoger” interrupciones que no tienen que ver nada con él.

g) Dado que para ambos tipos de núcleos se debe esperar a que se completen tanto las rutinas de interrupción de los dispositivos como las correspondientes a las interrupciones software de sistema antes de realizar el cambio de contexto involuntario, su mayor o menor duración no influye en el tiempo de respuesta y, por tanto, tampoco lo hace a la hora de valorar si el uso de un núcleo expulsivo es beneficioso.