Diseño de Sistemas Operativos. Junio de 2010.
Ejercicio de procesos y memoria

Los sistemas operativos de propósito general presentan ciertas restricciones que dificultan su aplicación en sistemas de tiempo real. Uno de los parámetros más importantes en este tipo de sistemas es el tiempo de respuesta, es decir, el tiempo que transcurre desde que ocurre un cierto evento físico (por ejemplo, la activación de un sensor) hasta que comienza/reanuda su ejecución el proceso que lo trata. En un sistema de tiempo real hay que acotar el **peor** tiempo de respuesta posible.

Para analizar este parámetro, considérese una aplicación de tiempo real formada por *P* procesos que ejecuta en un sistema con *N* niveles de interrupción y tal que cada rutina de interrupción de un dispositivo usa una interrupción software de sistema para diferir las operaciones menos urgentes asociadas a la misma, como, por ejemplo, el desbloqueo del proceso vinculado con esa interrupción. Se pretende calcular el **peor** tiempo de respuesta en una situación donde se produce una interrupción de máxima prioridad estando en ejecución el proceso de menor prioridad (*Pm*) y tal que dicha interrupción, en el contexto de su interrupción software diferida, desbloquea al proceso de máxima prioridad (*PM*).

NOTA: Para el cálculo del **peor** tiempo de respuesta posible se deben tener en cuenta todos los posibles eventos (llamadas al sistema, excepciones por fallos de página y todo tipo de interrupciones) y su máximo anidamiento, pero se supondrá que no se pueden producir dos interrupciones del mismo nivel en el intervalo de tiempo analizado.

a) Considere un S.O. de propósito general con un núcleo no expulsivo y responda a las siguientes cuestiones:

> **a1)** Especifique la traza correspondiente al **peor** tiempo de respuesta posible para el escenario planteado, identificando todas las activaciones del sistema operativo que se producen durante la misma.
>
> **a2)** Exprese mediante una fórmula cómo se calcula ese **peor** tiempo de respuesta para el escenario dado.
>
> **a3)** ¿Cambiaría en algo ese cálculo si la interrupción que causa el desbloqueo hubiera sido de mínima prioridad?
>
> **a4)** Suponga que los *N* procesos de la aplicación usan sólo dos llamadas al sistema: una que tiene una sola fase (es decir, no incluye ningún bloqueo) que dura 20 unidades de tiempo, y otra con un código cuya ejecución requiere 40 unidades de tiempo pero que se ejecuta en dos fases (es decir, incluye un bloqueo), la primera de 30 unidades de tiempo y la segunda de 10. ¿Cuál(es) de estos datos se incluiría(n) en la fórmula?
>
> **a5)** Considere que existen problemas de sincronización entre ciertas secciones del código de algunas llamadas al sistema y las rutinas de interrupción. ¿Con qué mecanismo resuelve el sistema operativo este tipo de problemas de sincronización? ¿Cómo afectaría el uso de este mecanismo a la fórmula calculada?

b) Considere un S.O. de propósito general con un núcleo expulsivo y responda a las siguientes cuestiones:

> **b1)** Resuelva todas las cuestiones del apartado anterior para este tipo de núcleo.
>
> **b2)** Considere que existen problemas de sincronización entre las llamadas al sistema que requieren secciones críticas de muy corta duración que no incluyen bloqueos. ¿Con qué mecanismo resuelve el sistema operativo este tipo de problemas de sincronización? ¿Cómo afectaría el uso de este mecanismo a la fórmula calculada? ¿Por qué no se ha considerado esta cuestión en el caso de un núcleo no expulsivo?

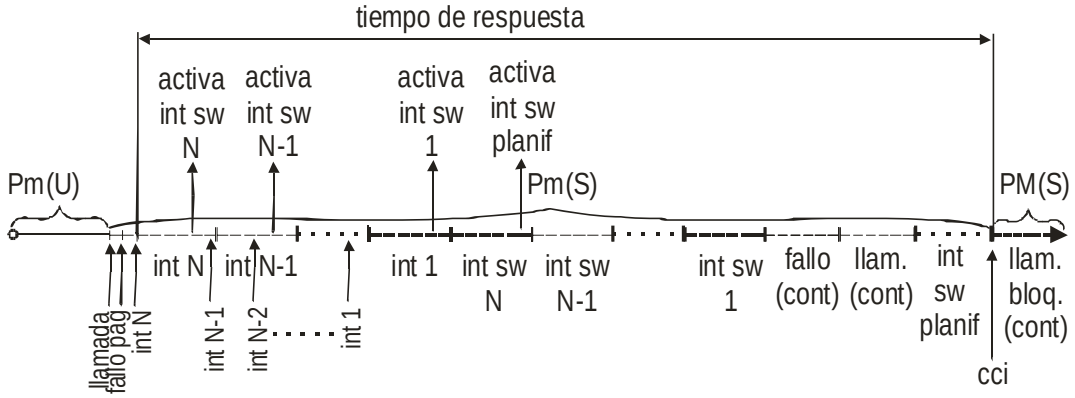
c) En un sistema de tiempo real, en el escenario planteado, la activación del proceso de máxima prioridad no debería haberse visto diferida por interrupciones de menor prioridad que la que causó el desbloqueo.

> **c1)** ¿Qué dificultades presenta conseguir ese requisito en un sistema operativo de propósito general?
>
> **c2)** Plantee de forma genérica una posible solución que permita satisfacer este requisito en un sistema operativo de propósito general.

Solución

a1) El tiempo de respuesta abarcará el intervalo que transcurre desde que se produce la interrupción de máxima prioridad hasta que se reactiva la ejecución del proceso *PM* que responde a la misma, es decir, hasta que se completa el cambio de contexto de *Pm* a *PM*. Al tratarse de la interrupción de mayor prioridad, su tratamiento comenzará de forma inmediata, pero el tiempo que pasa hasta que se reactiva la ejecución del proceso va a depender de qué eventos se produzcan en el sistema, siendo el peor caso posible cuando haya un mayor nivel de anidamiento.

A continuación, se muestra una traza, de las múltiples posibles, que corresponde a ese peor escenario de tiempo de respuesta.



En la traza planteada la interrupción de máxima prioridad llega en el momento que *Pm* ha hecho una llamada al sistema, tal que justo al principio de su ejecución se ha producido un fallo de página al acceder al mapa de usuario para obtener o depositar información, y se está en este instante al inicio del tratamiento de ese fallo de página.

Al tratarse de una interrupción de máxima prioridad, se activa inmediatamente su tratamiento (a no ser que estuvieran las interrupciones inhabilitadas, una situación que se estudiará en el apartado *a5*). Como plantea el enunciado, la rutina de interrupción realiza las labores más urgentes y activa una interrupción software de sistema para diferir el resto de las operaciones (entre ellas el desbloqueo de *PM*). El peor escenario posible se producirá cuando durante la rutina de interrupción se recibe al menos una interrupción de otro dispositivo, cuyo tratamiento comenzará justo al completarse la rutina de interrupción de máxima prioridad. Da igual cuál sea el orden de llegada de las interrupciones para que se produzca esa situación de peor tiempo de respuesta posible con tal de que se activen consecutivamente las *N* rutinas de interrupción presentes en el sistema.

Dado que cada rutina de interrupción ha activado su interrupción software de sistema asociada, al concluirse la secuencia de rutinas de interrupción, se producirá la correspondiente secuencia de rutinas de interrupción software de sistema. La primera de estas rutinas desbloquea a *PM* y, al detectar que ese proceso es más prioritario que el actual, activa una interrupción software de planificación. Como se trata de un núcleo no expulsivo, la activación de la rutina de tratamiento de esa interrupción software de planificación debe esperar a que se complete todo el trabajo en modo sistema que está llevándose a cabo. Por tanto, en primer lugar, se ejecutarán todas las rutinas de interrupción software de sistema, y, a continuación, proseguirá la rutina de tratamiento de fallo de página. Esa rutina de fallo puede completarse, en cuyo caso continuará la ejecución de la llamada, o bloquearse si requiere realizar una operación en el disco, realizándose en este caso un cambio de contexto voluntario a *PM*. El peor caso correspondería a que la rutina de fallo se complete sin bloqueo (un fallo de página que no lea ni escriba en el dispositivo de paginación) y se reanude la ejecución de la llamada. Ésta a su vez podría bloquearse, haciéndose entonces un cambio de contexto voluntario a *PM*, o completarse (que es la situación mostrada en la traza de la figura), activándose en ese momento la rutina de la interrupción software de planificación que lleva a cabo el cambio de contexto involuntario a *PM*.

Realmente, el peor escenario es todavía un poco más complicado: al reanudarse la llamada (sólo en el caso de que la rutina de fallo no haya requerido un bloqueo), podrían producirse nuevos fallos de página dentro de la misma, y si éstos no conllevan bloqueo, el cambio de contexto debería esperar a que se completen todos ellos, así como la llamada. El número de fallos de página dependerá de las características de cada llamada. Así, por ejemplo, una llamada de lectura o escritura podría causar tantos fallos como páginas ocupe el *buffer* de lectura/escritura especificado en la llamada. Lo más habitual es que estos fallos de página requieran leer la página de disco, con lo que no afectarían al tiempo de respuesta los fallos posteriores al haber un bloqueo. Sin embargo, puede ocurrir que se trate de una página de una región anónima sobre la que nunca se ha escrito (eso sería bastante raro en el caso de una llamada de tipo escritura en un fichero, pero sí podría darse en una de tipo lectura de fichero) o que la página requerida se encuentre en la caché de páginas, y en esos casos no habría que leerla del disco de paginación. Si todos los fallos que produce una llamada tienen esta característica de no bloqueo y la propia llamada también, todos ellos afectarán al tiempo de respuesta. Para acotar esta situación, en un núcleo no expulsivo el programador de una llamada al sistema donde potencialmente pudiera darse este problema podría incluir explícitamente en ciertos puntos del código de la misma donde las condiciones fueran adecuadas (por ejemplo, en cada iteración del bucle en una llamada de lectura o escritura de un fichero, en un punto donde se estuviera ejecutando siempre en el contexto de una llamada al sistema, con las interrupciones habilitadas y sin estar en posesión de un *spinlock* en el caso de un multiprocesador) una comprobación de si hay un proceso más prioritario listo para ejecutar y, en caso afirmativo, hacer un cambio de contexto al mismo.

a2) En la fórmula del peor tiempo de respuesta hay que utilizar el máximo tiempo de ejecución que puede conllevar el tratamiento de cada uno de los eventos implicados en la misma. En el caso de que durante el tratamiento de un evento pudiera haber un bloqueo (por tanto, sólo en una llamada o en un fallo de página), sólo se tendría en cuenta el tiempo hasta que se produce el mismo.

Así, en el cálculo del peor tiempo de respuesta, en lo referente a la influencia de las llamadas al sistema, que es uno de los eventos más críticos pues generalmente su duración es significativamente mayor que la de las rutinas de interrupción, habría

que usar como peor tiempo posible la duración de la fase más larga entre las de todas las llamadas al sistema, siendo una fase el tiempo que transcurre hasta que la llamada se bloquea o completa.

En cuanto a los fallos de página, como se explicó previamente, el peor caso ocurre cuando no incluyen un bloqueo, puesto que en caso de haberlo, no afectaría al tiempo de respuesta la finalización de la llamada al sistema ni otros posibles fallos de página posteriores dentro de la llamada. Nótese que habría que determinar cuál es el número máximo de fallos de página sin bloqueo que puede causar una fase de una llamada al sistema (realmente, para ser precisos, habría que tener en cuenta los dos factores involucrados, mayor duración de una fase de una llamada y número máximo de fallos sin bloqueo que puede generar, y buscar para qué fase de qué llamada el tiempo total es mayor).

Pasando ya a la fórmula, el peor tiempo de respuesta (*PTR*) incluiría los siguientes términos:

- el tiempo máximo que consume cada rutina de interrupción (*TIDi* para el nivel *i*)
- el tiempo máximo que consume cada rutina de interrupción software de sistema correspondiente (*TISSi* para el nivel *i*)
- la máxima duración de la fase más larga de una llamada al sistema (*TLL*)
- la máxima duración de un tratamiento de fallo de página sin bloqueo (*TFP*), y el número máximo de fallos de página sin bloqueo que se pueden producir durante una fase de una llamada al sistema (*NFP*)
- el tiempo máximo que consume la rutina de interrupción software de planificación (*TISP*)
- el tiempo máximo que conlleva el cambio de contexto

La fórmula resultante sería la siguiente:

$$PTR = PTLL + NFP * TFP + \sum TIDi + \sum TISSi + TISP + TCC$$

a3) Como se puede apreciar en la fórmula calculada, el peor tiempo de respuesta no variaría en el caso de que el desbloqueo lo produjera la interrupción de menor prioridad dado que la reactivación del proceso debe esperar a que se complete todo el anidamiento de interrupciones con independencia de cuál sea la interrupción que produce el desbloqueo.

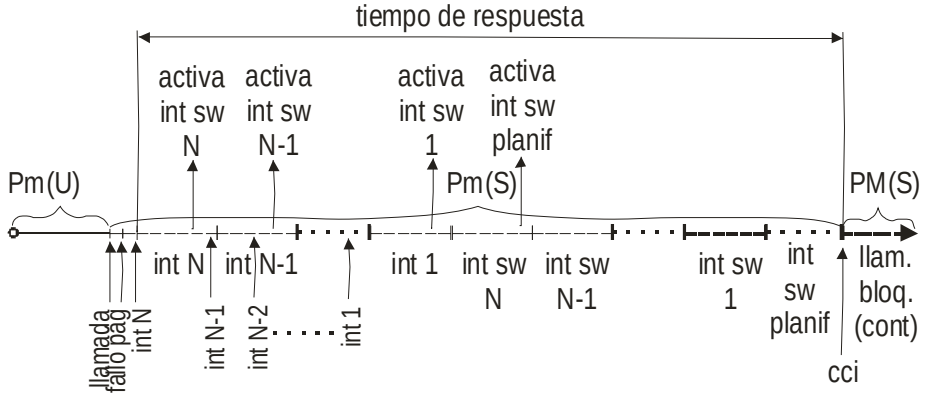
a4) Como se ha explicado previamente, para el cálculo del peor tiempo de respuesta posible se usa la duración de la fase más larga, que en el ejemplo planteado es de 30 unidades de tiempo.

a5) El mecanismo para evitar problemas de sincronización entre una llamada al sistema y una rutina de interrupción en un sistema monoprocesador es inhibir la interrupción conflictiva en la sección de código de la llamada al sistema afectada por la misma.

En principio, puede parecer que este tiempo de inhibición de interrupciones dentro de la llamada puede afectar al peor tiempo de respuesta ya que si coincide la activación de la interrupción de máxima prioridad con ese intervalo se retrasará consecuentemente. Sin embargo, se va a mantener el mismo peor tiempo de respuesta ya que la suma acumulada de tiempos de tratamiento de eventos es la misma puesto que el cambio de contexto no se lleva a cabo hasta que se termina toda la actividad en modo sistema (dicho de otra forma, el tiempo durante el que está inhabilitada la interrupción ya se ha tenido en cuenta dentro del tiempo total de la llamada).

b1.1) Al tratarse de un núcleo expulsivo, la interrupción software de planificación sólo espera a que se completen las rutinas de interrupción de los dispositivos y las rutinas de interrupción de sistema asociadas, no dejando que se reanude el tratamiento de eventos síncronos.

En consecuencia, la traza del peor escenario de tiempo de respuesta planteada en el primer apartado queda de la siguiente forma en un núcleo expulsivo:



Como se puede apreciar en la figura, la diferencia con respecto a la traza para un núcleo no expulsivo se produce al finalizar la secuencia de rutinas de interrupción software de sistema. En este caso, justo después de las mismas entra la interrupción

software de planificación que procede a llevar a cabo al cambio de contexto, mientras que en el núcleo no expulsivo, éste tenía que esperar hasta que se completaran, o bloquearan, los tratamientos de los eventos síncronos.

b1.2) Como consecuencia de lo explicado, la fórmula del peor tiempo de respuesta para este tipo de núcleo sería la siguiente:

$$PTR = \sum TIDi + \sum TISSi + TISP + TCC$$

b1.3) Como ocurría con el núcleo no expulsivo, el peor tiempo de respuesta no depende del nivel de la interrupción que causa el desbloqueo.

b1.4) No habría que incluir ninguno de esos tiempos puesto que la duración de las llamadas al sistema no influye en el peor tiempo de respuesta posible en este tipo de núcleo.

b1.5) Como ocurría con el núcleo no expulsivo, se usa el mecanismo de inhabilitar las interrupciones para evitar este tipo de problemas de sincronización. Sin embargo, en este tipo de núcleo este mecanismo afecta al tiempo de respuesta puesto que puede ocurrir que se hayan inhibido las interrupciones justo en el momento que se ha producido la interrupción, de manera que la activación de la rutina de interrupción tiene que esperar hasta que se rehabiliten. Por tanto, a la fórmula del peor tiempo hay que añadirle el tiempo máximo durante el que están inhabilitadas las interrupciones (*TNI*).

$$PTR = TNI + \sum TIDi + \sum TISSi + TISP + TCC$$

b2) El mecanismo utilizado para evitar problemas de sincronización en este tipo de núcleos para el caso de que la sección crítica involucrada sea muy breve y no incluya bloqueos es impedir las expulsiones (es decir, inhabilitar los cambios de contexto involuntarios inhibiendo el tratamiento de la interrupción software de planificación) durante el intervalo conflictivo.

Este factor influye en el peor tiempo de respuesta posible ya que en el caso de que se produzca la interrupción de máxima prioridad mientras están deshabilitadas las expulsiones, la interrupción software de planificación no se activará justo después de las rutinas de interrupción software de sistema, sino que antes se reanudará la ejecución de la llamada hasta que ésta vuelva a rehabilitar las expulsiones al final de la sección crítica. Nótese que en este caso no podría haber un fallo de página anidado con la llamada al sistema puesto que se trata de una sección crítica que no incluye bloqueos (en caso de que la sección crítica los incluyera o no fuera muy breve habría que haber usado un semáforo para conseguir la correcta sincronización).

Como consecuencia de este mecanismo habría que incorporar un nuevo sumando a la fórmula: el tiempo máximo durante el que están inhabilitadas las expulsiones (*TNE*).

$$PTR = TNE + TNI + \sum TIDi + \sum TISSi + TISP + TCC$$

En cuanto a los núcleos no expulsivos, no tiene sentido considerar este tiempo en los mismos puesto que en ellos no se pueden dar llamadas concurrentes.

c1) El modelo de operación de un sistema de propósito general asume que el tratamiento de cualquier interrupción, aunque sea de mínima prioridad, es más urgente que la ejecución de cualquier proceso, aunque sea de máxima prioridad. En un sistema de tiempo real esto no es así y, por tanto, hay que integrar de alguna forma las prioridades de las interrupciones y de los procesos.

En principio podría parecer que basta con integrar en un esquema único las prioridades de las interrupciones y de los procesos, y añadir un mecanismo al sistema operativo de propósito general que asegure que mientras ejecuta un proceso no se active el tratamiento de las interrupciones que se consideren menos urgentes que el mismo. Sin embargo, el asunto es más complicado. Para entenderlo, considere el siguiente ejemplo: estando activo el tratamiento de una interrupción de baja prioridad, se ve interrumpido por una de mayor prioridad que desbloquea un proceso que se ha definido como más urgente que la interrupción de baja prioridad. En ese ejemplo, se debería ceder el control a ese proceso sin retrasarle hasta que se complete la rutina de tratamiento de la interrupción de prioridad más baja (o interrupciones, en caso de haber anidamiento); pero si así lo hacemos, ¿quién, cómo y cuándo se completarán esas rutina(s) de interrupción inacabadas(s)? En el caso de un sistema de propósito general, no hay forma de hacerlo, porque se implementa un modelo en el que bajo ninguna circunstancia la ejecución de una rutina de interrupción queda inacabada para dar paso a otro proceso.

c2) Una posible solución para dotar al procesamiento de una interrupción de un contexto independiente que pueda ser interrumpido por la ejecución de un proceso para ser posteriormente completado es hacer que el tratamiento de la misma lo realice un proceso/*thread* de núcleo. Cada nivel de interrupción tendría asignado un proceso de núcleo, con su propia pila de núcleo como todos los procesos, de manera que cuando se produjera una interrupción de ese nivel se activaría el proceso de núcleo asociado. El propio mecanismo de planificación de procesos definiría un esquema integrado de niveles de prioridad que englobaría también a las interrupciones.
En el ejemplo planteado en el apartado anterior, el proceso desbloqueado tendrá más prioridad que el proceso de núcleo que ejecuta el tratamiento de la interrupción de baja prioridad, por lo que pasará a ejecutar inmediatamente. El tratamiento de dicha interrupción de baja prioridad se completará cuando el proceso de núcleo que la sirve sea el proceso listo en el sistema con mayor prioridad.