Diseño de Sistemas Operativos. Septiembre de 2010.
Ejercicio de procesos

Un amigo tuyo, Mr. Hype, está intentando convencerte de que recompiles la versión de Linux que tienes instalada en tus dos máquinas (una de uso personal en la que se ejecutan numerosas aplicaciones multimedia; otra dedicada exclusivamente a proporcionar un servicio web) de manera que se pase de un sistema con un núcleo no expulsivo a uno expulsivo. Para persuadirte, expone los argumentos que se recogen a continuación. Se pide que se analice cada uno de ellos y se explique razonadamente cuáles son verdaderos y cuáles falsos.

a) Utilizando un núcleo expulsivo el sistema será más fiable (es decir, se reducirá la frecuencia de *cuelgues* en el sistema).

b) Usando un núcleo expulsivo el sistema será más eficiente, en el sentido de que se realizará más trabajo útil por unidad de tiempo.

c) Con un núcleo expulsivo la duración de las llamadas al sistema no influirá en el tiempo de respuesta de los procesos.

d) Con un núcleo expulsivo la duración de las rutinas de interrupción no influirá en el tiempo de respuesta de los procesos.

e) Tu amigo te explica que este cambio será especialmente beneficioso para tu máquina de uso personal.

f) Como último argumento, tu amigo te ha explicado que con un núcleo expulsivo se mantendrá básicamente el número de cambios de contexto que se producen en el sistema. Para abordar esta cuestión, se plantea previamente analizar las siguientes trazas:

1. Considere una situación donde un proceso de baja prioridad *P* está realizando una llamada al sistema no bloqueante y durante la misma se produce una interrupción de un dispositivo *A* que desbloquea a un proceso de mayor prioridad *Q*, que cuando reanuda su ejecución realiza unos cálculos y termina. Especifique la traza de ejecución para cada tipo de núcleo, identificando **todas** las activaciones del sistema operativo y los cambios de contexto, distinguiendo entre voluntarios e involuntarios, y comparando el número de ellos que se producen para cada tipo de núcleo.

2. Repita el apartado previo suponiendo que la llamada al sistema que realiza *P* incluye una operación de entrada/salida sobre el dispositivo *B* y es, por tanto, bloqueante. Suponga que la interrupción del dispositivo *A* (la que desbloquea a *Q*) es anterior al bloqueo de *P* y que la interrupción del dispositivo *B* (la que desbloquea a *P*) no se produce en el intervalo de tiempo analizado.

3. Repita el **primer** apartado (en el que *P* realiza una llamada no bloqueante) pero suponiendo ahora que durante la llamada se produce una primera interrupción, que desbloquea a un proceso *Q* de prioridad media, y, un cierto tiempo después de que se haya completado la ejecución de las actividades asociadas a esta primera interrupción pero un poco antes de que se vuelva a modo usuario, llega una segunda interrupción que desbloquea a un proceso *R* de prioridad alta. Ambos procesos, una vez desbloqueados, realizan una fase de cálculo y completan su ejecución.

4. En esta última traza, como en la primera, existe un proceso de baja prioridad *P* realizando una llamada al sistema no bloqueante. Sin embargo, en este caso, hay una sección del código de la llamada que presenta problemas de condiciones de carrera si varios procesos la ejecutan de forma concurrente y, por tanto, se debe asegurar que se ejecuta de forma exclusiva. Como ocurría en la primera traza, mientras *P* ejecuta la llamada, más concretamente, la sección reseñada, se produce una interrupción de un dispositivo que desbloquea a un proceso de mayor prioridad *Q*, que cuando reanuda su ejecución invoca la misma llamada al sistema y, una vez completada ésta, termina su ejecución.

5. A partir de las trazas anteriores, explique qué diferencias hay entre estos dos tipos de núcleos en lo que se refiere al número de cambios de contexto generados, distinguiendo entre voluntarios e involuntarios.

Solución

a) Hay que resaltar, en primer lugar, algo que puede parecer obvio: si el código de un sistema operativo fuera perfecto, daría igual, en lo que se refiere a fiabilidad, qué tipo de organización y modo de operación interno tuviera (expulsivo o no expulsivo, monolítico o micronúcleo,...). Sin embargo, la experiencia en la construcción de grandes sistemas software ha mostrado que es prácticamente imposible, al menos con las tecnologías de las que hemos dispuesto hasta el momento, crear un sistema de estas características que esté libre de errores. Sólo nos queda asumir este hecho e intentar usar técnicas que nos permitan limitar dentro de lo posible la tasa de errores en el sistema.
Una de las fuentes principales de errores en un sistema operativo, dado su alto grado de concurrencia y asincronismo, es la sincronización de las distintas actividades que se llevan a cabo en un determinado momento. Un núcleo no expulsivo limita su concurrencia interna, al no permitir que se ejecuten concurrentemente llamadas al sistema, reduciendo drásticamente sus problemas de sincronización interna. Un núcleo expulsivo, sin embargo, en aras a proporcionar un buen tiempo de respuesta, sí permite la ejecución concurrente de llamadas, por lo que requiere la inclusión de mecanismos de sincronización para evitar los problemas de condiciones de carrera que se producen al ejecutarse concurrentemente llamadas al sistema. Dada la complejidad de estos esquemas de sincronización, son propensos a fallos, lo que hace que este tipo de núcleos expulsivos tengan a priori un grado de fiabilidad menor.

b) La labor de un sistema operativo de propósito general es ejecutar las aplicaciones activas en el sistema y gestionar el hardware incluido en el mismo. Un sistema operativo multiprogramado debe repartir de forma transparente los recursos existentes en el sistema entre las distintas actividades presentes en el mismo en un instante dado, sin que éstas puedan interferirse de manera errónea. Esta multiplexación de recursos, en la que se incluye las operaciones de cambio de contexto, no es trabajo útil propiamente dicho y conlleva una cierta sobrecarga.
Un núcleo de tipo expulsivo persigue mejorar el tiempo de respuesta del sistema permitiendo que un proceso pueda ser expulsado cuando se desbloquea otro más prioritario incluso aunque el primero esté en ese momento realizando una llamada al sistema, a diferencia del núcleo no expulsivo que sólo posibilita la expulsión cuando el proceso está en modo usuario. Como se analizará más adelante, este comportamiento conlleva un mayor número de cambios de contexto y, por tanto, una mayor sobrecarga y un menor nivel de eficiencia medio.

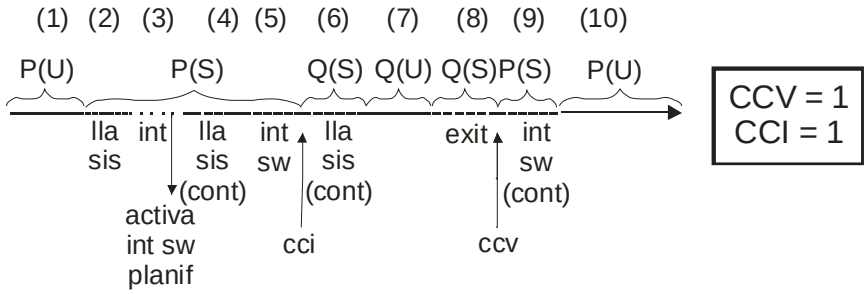
c) Con un núcleo expulsivo, si se desbloquea un proceso mientras otro de menor prioridad está realizando una llamada al sistema, se expulsa al proceso menos prioritario que completará más tarde esa llamada, cuando se le vuelva a asignar el procesador. Por tanto, en un núcleo expulsivo, la duración de las llamadas al sistema no influye en el tiempo de respuesta (o de activación) de los procesos.
Esto contrasta con lo que ocurre en un núcleo no expulsivo, donde el cambio de contexto involuntario no se lleva a cabo hasta que el proceso de menor prioridad complete la llamada, por lo que la duración de las llamadas al sistema repercute en el tiempo de respuesta del sistema.

d) En un sistema operativo de propósito general, con independencia del tipo de núcleo que se utilice, las interrupciones son siempre más prioritarias que cualquier proceso. Por tanto, si se desbloquea un proceso de mayor prioridad que el que está en ejecución durante el tratamiento de una o más interrupciones, la expulsión debe diferirse hasta que se completen los tratamientos de interrupción pendientes.
En consecuencia, para ambos tipos de núcleos, la duración de las rutinas de interrupción repercute en el tiempo medio de respuesta del sistema.

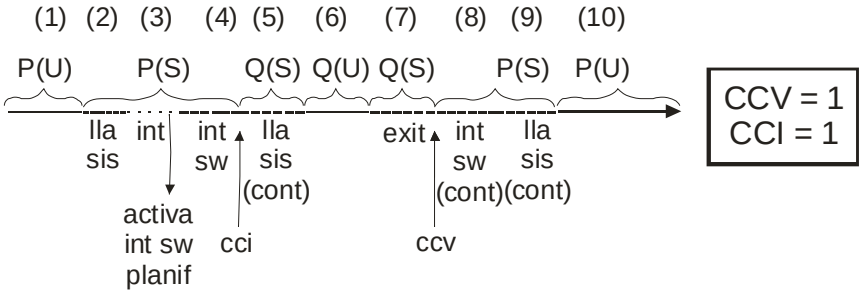
e) El principal beneficio de usar un núcleo expulsivo es mejorar el tiempo de respuesta de los procesos. Esta característica es especialmente necesaria cuando las aplicaciones que se van a usar en el sistema tienen requisitos de tiempo real (no crítico, por supuesto), como ocurre con las aplicaciones multimedia, para los cuales un retraso puntual en el tiempo de respuesta puede conllevar una mala calidad en la reproducción del medio correspondiente.
En sistemas que se dedican a ejecutar aplicaciones que no tengan este tipo de requisitos, como ocurre con la máquina servidora planteada en el enunciado, puede ser más interesante usar un núcleo no expulsivo que ofrece mayor fiabilidad y una menos sobrecarga.

f1) La siguiente figura muestra la traza de ejecución planteada en el enunciado tanto para un núcleo expulsivo como para uno no expulsivo:

versión no expulsiva



versión expulsiva



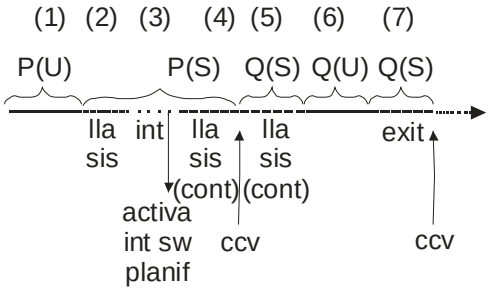
En la traza correspondiente al núcleo no expulsivo, se puede observar que mientras el proceso P está realizando una llamada al sistema (2), se produce una interrupción del dispositivo A. En el tratamiento de esta interrupción (3), se desbloquea al proceso Q y, al detectar que es más prioritario que el proceso en ejecución, se activa una interrupción software de planificación dirigida al proceso en ejecución cuyo tratamiento debe esperar hasta que se complete la llamada. Por tanto, al finalizar la rutina de interrupción del dispositivo, prosigue la llamada al sistema (4). Cuando ésta se completa, comienza el tratamiento de la interrupción software de planificación (5), que realiza el cambio de contexto involuntario de P a Q. Este segundo proceso continuará su ejecución en el punto donde la dejó, que será en el código del cambio de contexto voluntario dentro de una llamada al sistema que opera sobre el dispositivo A, y completará dicha llamada al sistema (6) prosiguiendo en modo usuario (7). Finalmente, el proceso Q completa su ejecución e invoca la llamada al sistema correspondiente disponible para tal fin, en cuyo tratamiento (8) se hace un cambio de contexto voluntario al proceso P. Este proceso proseguirá su ejecución en el punto donde la abandonó, es decir, dentro de la rutina de tratamiento de la interrupción software de planificación (9). Al completar esa rutina, vuelve a ejecutar en modo usuario (10).

En el caso de un núcleo expulsivo, la diferencia aparece a partir del punto 3. En este caso, la interrupción software de planificación es de tipo expulsiva, por lo que al completarse el tratamiento de la interrupción del dispositivo A, entra a ejecutar la rutina de tratamiento de la interrupción software de planificación (4), sin dejar que P complete la llamada al sistema. En esta rutina de tratamiento se hace el cambio de contexto involuntario de P a Q y, a continuación, ocurre la misma secuencia de eventos que en el caso de un núcleo no expulsivo: Q completa la llamada en la que estaba bloqueado (5), ejecuta en modo usuario hasta que completa su ejecución (6) y llama al sistema para notificarlo, realizándose en el tratamiento de esta llamada de finalización (7) el cambio de contexto voluntario al proceso P. Como en la traza previa, P prosigue su ejecución en la rutina de tratamiento de la interrupción software de planificación donde fue expulsado (8) pero, a diferencia del caso previo, cuando se completa este tratamiento, P prosigue con la llamada al sistema inacabada (9), completándola y volviendo a ejecutar en modo usuario (10).

Nótese que para el ejemplo planteado se produce el mismo número de cambios de contexto para los dos núcleos: un cambio voluntario y uno involuntario.

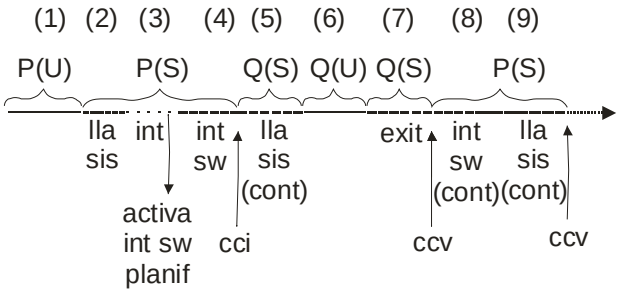
f2) La siguiente figura muestra la traza de ejecución planteada en el enunciado tanto para un núcleo expulsivo como para uno no expulsivo:

versión no expulsiva



CCV = 2
CCI = 0

versión expulsiva

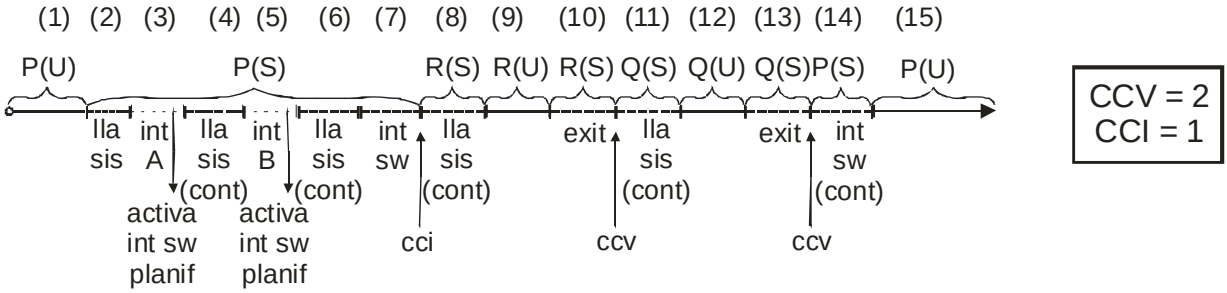


CCV = 2
CCI = 1

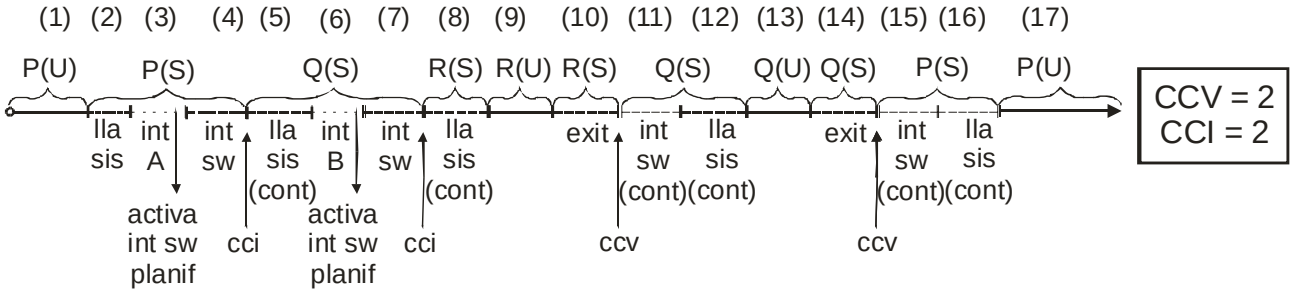
La traza correspondiente al núcleo no expulsivo es igual a la descrita en el punto anterior para este mismo tipo de núcleo hasta el punto 4, que corresponde a la reanudación de la llamada. En este caso, tal como plantea el enunciado, la llamada es bloqueante y, por tanto, el proceso P no completa la llamada sino que realiza un cambio de contexto a Q como consecuencia del bloqueo. Nótese que como parte de las operaciones asociadas al bloqueo se desactivaría la interrupción software de planificación vinculada a este proceso puesto que ya no es necesaria (dicho de manera informal, queríamos expulsar a este proceso pero el mismo se ha ido voluntariamente antes). A continuación, con el proceso Q ocurre la misma secuencia de eventos que en las trazas previas: Q completa la llamada en la que estaba bloqueado (5), ejecuta en modo usuario hasta que completa su ejecución (6) y llama al sistema para notificarlo (7).
Por lo que se refiere al núcleo expulsivo, dado que la interrupción del dispositivo, con la consiguiente expulsión, se produce antes del bloqueo, la traza será igual a la descrita en el punto anterior para este mismo tipo de núcleo hasta el punto 9, que corresponde a la reanudación de la llamada al sistema de P. En este caso la llamada no se completa, sino que el proceso P se bloquea realizando un cambio de contexto voluntario.
En este segundo ejemplo, hay el mismo número de cambios de contexto voluntarios (2) en ambos tipos de núcleos. Sin embargo, en cuanto a cambios involuntarios, se produce uno para el núcleo expulsivo pero ninguno para el no expulsivo, puesto que el proceso que se tenía que expulsar se ha bloqueado previamente.

f3) La siguiente figura muestra la traza de ejecución planteada en el enunciado tanto para un núcleo expulsivo como para uno no expulsivo:

versión no expulsiva



versión expulsiva



En la traza correspondiente al núcleo no expulsivo, se puede observar que mientras el proceso P está realizando una llamada al sistema (2), se produce una interrupción del dispositivo A. En el tratamiento de esta interrupción (3), se desbloquea al proceso Q y, al detectar que es más prioritario que el proceso en ejecución, se activa una interrupción software de planificación dirigida al proceso en ejecución cuyo tratamiento deberá esperar hasta que se complete la llamada. Por tanto, al finalizar la rutina de interrupción del dispositivo, prosigue la llamada al sistema (4) y se produce una interrupción del dispositivo B. En el tratamiento de esta segunda interrupción (5), se desbloquea al proceso R y, al detectar que es más prioritario que el proceso en ejecución, se activa una segunda interrupción software de planificación dirigida al mismo proceso (nótese que no es necesario que se encole este tipo de eventos). Nuevamente, se reanuda la llamada al sistema (6) y cuando ésta se completa, comienza el tratamiento de la interrupción software de planificación (7), que realiza el cambio de contexto involuntario de P a R, al ser éste el proceso más prioritario. Este proceso continuará su ejecución en el punto donde la dejó, que será en el código del cambio de contexto voluntario dentro de una llamada al sistema que opera sobre el dispositivo B, y completará dicha llamada al sistema (8) prosiguiendo en modo usuario (9). Finalmente, el proceso Q completa su ejecución e invoca la llamada al sistema correspondiente disponible para tal fin, en cuyo tratamiento (10) se hace un cambio de contexto voluntario al proceso Q. A continuación, ocurre la misma secuencia de eventos que en el caso del proceso R: Q completa la llamada en la que estaba bloqueado (11), ejecuta en modo usuario hasta que completa su ejecución (12) y llama al sistema para notificarlo, realizándose en el tratamiento de esta llamada de finalización (13) el cambio de contexto voluntario al proceso P. Este proceso proseguirá su ejecución en el punto donde la abandonó, es decir, dentro de la rutina de tratamiento de la interrupción software de planificación (14). Al completar esa rutina, vuelve a ejecutar en modo usuario (15).

En el caso de un núcleo expulsivo, la diferencia aparece a partir del punto 3. En este caso, la interrupción software de planificación es de tipo expulsiva, por lo que al completarse el tratamiento de la interrupción del dispositivo A, entra a ejecutar la rutina de tratamiento de la interrupción software de planificación (4), sin dejar que P complete la llamada al sistema. En esta rutina de tratamiento se hace el cambio de contexto involuntario de P a Q. El proceso Q continuará su ejecución en el punto donde la dejó, que será en el código del cambio de contexto voluntario dentro de una llamada al sistema que opera sobre el dispositivo A (5). En ese instante se produce una interrupción del dispositivo B. En el tratamiento de esta segunda interrupción (6), se desbloquea al proceso R y, al detectar que es más prioritario que el proceso en ejecución, se activa una interrupción software de planificación dirigida al proceso en ejecución. Cuando se completa el tratamiento de la interrupción del dispositivo B, entra a ejecutar la rutina de tratamiento de la interrupción software de planificación (7), sin dejar que Q complete la llamada al sistema. En esta rutina de tratamiento se hace el cambio de contexto involuntario de Q a R. El proceso R completa la llamada al sistema donde se quedó bloqueado (8), ejecuta en modo usuario hasta que

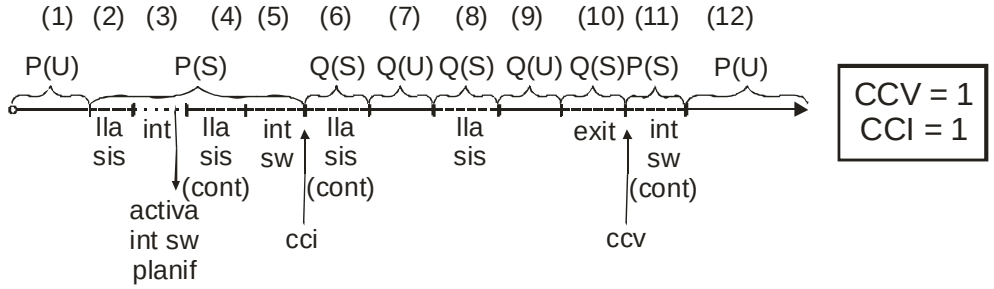
completa su ejecución (9) y llama al sistema para notificarlo, realizándose en el tratamiento de esta llamada de finalización (10) el cambio de contexto voluntario al proceso Q. Este proceso proseguirá su ejecución en el punto donde la abandonó, es decir, dentro de la rutina de tratamiento de la interrupción software de planificación (11), y cuando termine este tratamiento, proseguirá con la llamada al sistema inacabada (12), completándola y volviendo a ejecutar en modo usuario (13). Finalmente, el proceso Q completa su ejecución y llama al sistema para notificarlo, realizándose en el tratamiento de esta llamada de finalización (14) el cambio de contexto voluntario al proceso P. Este proceso proseguirá su ejecución dentro de la rutina de tratamiento de la interrupción software de planificación (15), completando a continuación la llamada al sistema inacabada (16), y volviendo a ejecutar en modo usuario (17).
En este tercer ejemplo, vuelve a haber el mismo número de cambios de contexto voluntarios (2) en ambos tipos de núcleos. Sin embargo, con respecto a los cambios involuntarios, se producen más en el núcleo expulsivo (2) que en el no expulsivo (1).
Otro aspecto que se puede apreciar comparando estas dos trazas es la sobrecarga que introduce el carácter expulsivo: en la traza expulsiva hay 17 pasos mientras que en la no expulsiva se producen 15.

f4) Antes de pasar a analizar las trazas hay que revisar qué técnicas se utilizan para resolver los problemas de sincronización que se producen cuando se ejecutan concurrentemente llamadas al sistema.
En el caso de un núcleo no expulsivo no es necesario utilizar ningún mecanismo para afrontar este tipo de problemas puesto que no se pueden producir en este tipo de núcleos al no permitirse la ejecución concurrente de llamadas.
En los núcleos expulsivos sí se permite la ejecución concurrente de llamadas y, por tanto, se requiere usar algún tipo de técnica. Básicamente, se utilizan dos mecanismos:

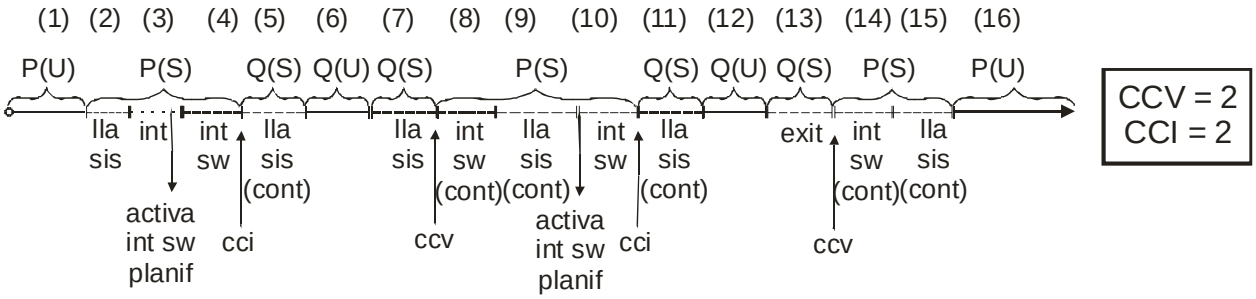
- Si la sección crítica afectada es muy breve y no incluye ningún bloqueo, se puede inhibir la interrupción software de planificación durante la misma, eliminando con ello la posibilidad de ejecución concurrente en esa sección.
- En los casos restantes, que son la mayoría, se usa un mecanismo de tipo semáforo para asegurar la exclusión mutua durante la sección crítica.

Dado que la segunda solución es la más habitual y que conlleva una traza de ejecución más compleja, vamos a optar por usarla a la hora de plantear la traza correspondiente al núcleo expulsivo.
Una vez revisadas las técnicas de sincronización, pasamos a detallar las trazas del ejemplo planteado en el enunciado. La siguiente figura muestra la traza de ejecución tanto para un núcleo expulsivo como para uno no expulsivo:

versión no expulsiva



versión expulsiva



La traza correspondiente al núcleo no expulsivo es igual a la descrita en el primer ejemplo para este mismo tipo de núcleo puesto que, como se acaba de explicar, no es necesario el uso de ningún mecanismo de sincronización. La única diferencia con esa traza inicial proviene de que en este caso el enunciado plantea que el proceso Q realiza una llamada al sistema durante su ejecución (8). Observe que en la traza se puede comprobar que en este tipo de núcleos no se producen llamadas concurrentes.
En el caso de un núcleo expulsivo, la traza es igual a la presentada en el primer ejemplo para este mismo tipo de núcleo hasta el punto 6, que corresponde al momento cuando Q está ejecutando en modo usuario. En ese momento, tal como plantea el enunciado, el proceso Q realiza la misma llamada al sistema que estaba ejecutando P cuando fue expulsado. Más concretamente, en el momento de la expulsión, P estaba ejecutando una sección crítica dentro de la llamada, por lo que estaba en posesión del semáforo que protege esa sección crítica. Por tanto, cuando Q ejecuta el código de esa misma llamada (7) e intenta obtener ese semáforo, se queda bloqueado haciendo un cambio de contexto voluntario de Q a P. P prosigue su ejecución en la rutina de tratamiento de la interrupción software de planificación donde fue expulsado (8) y, a continuación, cuando se completa este tratamiento, P continúa con la llamada al sistema inacabada (9). Durante el transcurso de la misma, P libera el semáforo cuando termina la sección crítica desbloqueando al proceso Q. En el momento en el que se detecta que el proceso desbloqueado es más prioritario que el proceso en ejecución, se activa una interrupción software de planificación dirigida al proceso en ejecución, cuya rutina de tratamiento entra a ejecutar (10) en ese momento al tratarse de un núcleo expulsivo, sin dejar que P complete la llamada al sistema. En esta rutina de tratamiento se hace el cambio de contexto involuntario de P a Q. El proceso Q completa la llamada (11) y, a continuación, ejecuta en modo usuario hasta que completa su ejecución (12) y llama al sistema para notificarlo. En el tratamiento de esta llamada de finalización (13) se realiza el cambio de contexto voluntario al proceso P. P prosigue su ejecución en la rutina de tratamiento de la interrupción software de planificación (14), completando a continuación la llamada al sistema inacabada (15), y volviendo a ejecutar en modo usuario (16).
En este ejemplo, en el núcleo expulsivo hay más cambios de contexto involuntarios (2 frente a 1) y también más voluntarios (2 frente a 1), como consecuencia de la necesidad de usar mecanismo de sincronización.

g) Revisando las trazas presentadas en los apartados previos, se puede concluir que en un núcleo expulsivo hay una tendencia a que se produzcan más cambios de contexto que en uno no expulsivo debido básicamente a tres circunstancias, todas ellas relacionadas con el hecho de que en un núcleo expulsivo se puede producir la expulsión de un proceso en medio de una llamada al sistema mientras que en uno no expulsivo esto no es posible:

- Un núcleo expulsivo se puede *ahorrar* un cambio de contexto involuntario cuando se da una situación en la que se debe expulsar un proceso que está realizando una llamada al sistema bloqueante (segundo ejemplo), ya que el proceso deja voluntariamente el procesador antes de ser expulsado.
- En situaciones donde estando en ejecución un proceso de baja prioridad realizando una llamada al sistema se produce una secuencia de eventos que van desbloqueando a procesos de prioridad creciente (tercer ejemplo), en un núcleo no expulsivo puede ocurrir que se produzcan todos ellos durante la llamada al sistema y, por tanto, al final de la misma, sólo haya un cambio de contexto involuntario al proceso de mayor prioridad de la secuencia. Sin embargo, en un núcleo expulsivo se producirá en este caso una secuencia de cambios de contexto involuntarios.
- La necesidad de usar mecanismos de sincronización para evitar condiciones de carrera durante la ejecución concurrente de llamadas (cuarto ejemplo) puede conllevar cambios de contexto adicionales, tanto de tipo voluntario como involuntario.

En cualquier caso, esta tendencia a tener un número mayor de cambios de contexto es el precio que hay que pagar para conseguir un sistema con un mejor tiempo de respuesta.