DISEÑO DE SISTEMAS OPERATIVOS 4º. Febrero de 2004

Ejercicio 1

Responda razonadamente a las siguientes preguntas sobre gestión de memoria y procesos.

a) La TLB (*Translation Lookaside Buffer*) agiliza el proceso de traducción de direcciones manteniendo información de las más recientes. La gestión de la TLB la realiza directamente la MMU del procesador sin intervención del S.O. Sin embargo, el S.O. sí debe realizar cierta gestión de la TLB: para asegurar la coherencia, debe invalidar entradas de la TLB cuando su información deje de ser correcta.

Considérese una TLB convencional, que no incluye ninguna información de proceso, tal que cada entrada contiene la información habitual: número de página, número de marco, permisos de acceso y bits de referencia y de modificado. Supóngase que el procesador incluye sólo dos instrucciones para que el S.O. pueda manipular la TLB: una para invalidar la entrada correspondiente a una determinada página y otra para invalidar completamente el contenido de la TLB. Analice de forma razonada si en cada una de las siguientes operaciones, el S.O. haría uso de alguna de estas instrucciones, especificando, en el caso de que se invalide una entrada, a qué página correspondería:

1. Hay un cambio de contexto entre dos *threads* de distinto proceso.
2. Hay un cambio de contexto entre dos *threads* del mismo proceso.
3. Se crea una nueva región en el mapa del proceso.
4. Se elimina una región del mapa del proceso.
5. Cambia el tamaño de una región del mapa del proceso. Analice separadamente el caso de aumento de tamaño y el de disminución.
6. Se marca como duplicada una región (por ejemplo, en el tratamiento de una región privada que forma parte de una llamada *fork*).
7. Hay un fallo de página que encuentra un marco libre.
8. Hay un fallo de página que no encuentra marcos libres. El sistema usa el algoritmo de reemplazo del reloj y se encuentra que la primera página candidata a ser expulsada tiene tanto el bit de referencia como el de modificado a 1, mientras que la segunda tiene ambos bits a 0.
9. Hay un fallo debido al *copy-on-write* y el contador de uso de la página es mayor que 1.
10. Hay un fallo debido al *copy-on-write* y el contador de uso es igual a 1.

b) Algunos procesadores, en vez de manejar las dos pilas habituales (de usuario y de sistema), gestionan tres pilas independientes (o sea, tres registros de puntero de pila, SP) que operan de la siguiente forma:

- Un puntero de pila de usuario (USP), usado por el procesador cuando está en modo usuario.
- Un puntero de pila de sistema (SSP), usado por el procesador cuando está en modo sistema debido a la ocurrencia de alguno de los siguientes eventos: llamada al sistema, fallo de página (por simplicidad, no se tratan el resto de las excepciones) e interrupción software.
- Un puntero de pila de interrupción (ISP), usado por el procesador cuando está en modo sistema debido a la ocurrencia de una interrupción externa de un dispositivo.

Dadas estas características, responda razonadamente a las siguientes cuestiones:

1. Analice si es posible que todos los procesos usen la misma zona de memoria como pila de sistema. ¿Y cómo pila de interrupción?
2. Explique cuál sería el contenido de las diferentes pilas del proceso justo en el momento en el que el proceso cede el procesador en un cambio de contexto voluntario. Plantee una situación con el máximo anidamiento posible. Analice si influye en la respuesta anterior (y, en caso afirmativo, cómo) si se trata de un kernel expulsivo o no.
3. Repita el ejercicio anterior en el caso de un cambio de contexto involuntario.
4. ¿Cuál sería el tamaño máximo de cada pila? Analice cómo influiría en este cálculo si el kernel es expulsivo o no.

Solución

a) Antes de responder a las distintas cuestiones sobre la repercusión en la TLB de diferentes operaciones vinculadas con la gestión de memoria, es conveniente explicar a qué se debe la necesidad de que el sistema operativo tenga que intervenir en la gestión de la TLB en determinadas ocasiones.

La TLB actúa como una cache de las tablas de páginas, manteniendo información sobre las últimas traducciones realizadas por la MMU. Como ocurre con cualquier otro tipo de cache (por ejemplo, la cache de memoria), puede haber problemas de coherencia si se modifica directamente el nivel superior de la jerarquía, que, en este caso, correspondería con la tabla de

páginas. Por tanto, cada vez que el sistema operativo modifique algún campo de una entrada de la tabla de páginas que potencialmente pueda estar incluida en la TLB, debe solicitar la invalidación de dicha entrada en la TLB, que sólo se producirá, evidentemente, si dicha entrada está incluida en la TLB. A continuación, se analizan los distintos casos planteados:

1. Cuando hay un cambio de contexto entre dos *threads* de distintos procesos, al tratarse de una TLB que no incluye información de proceso, es necesario que el sistema operativo **invalide completamente** el contenido de la TLB, ya que reflejaba traducciones referentes al proceso que deja al procesador. De hecho, como se verá en el siguiente caso, el coste de esta operación de invalidación es el principal factor que marca la notable diferencia entre el tiempo que requiere un cambio de contexto entre *threads* de distintos procesos y del mismo.
2. Si hay un cambio de contexto entre dos *threads* del mismo proceso, dado que comparten el mismo mapa de memoria, el sistema operativo **no tiene que realizar ninguna operación de invalidación**. Nótese que, aunque cada *thread* tenga su propia pila, todos comparten el mapa de memoria.
3. La creación de una región **no implica ninguna invalidación**, ya que se trata de un sistema de paginación por demanda.
4. Si se elimina una región del mapa del proceso, es necesario que el sistema operativo **invalide las entradas correspondientes a esa región** que pueda haber en la TLB, puesto que esas traducciones ya no son correctas.
5. El caso del aumento en el tamaño de una región es similar al analizado en el segundo punto (la creación de una región). Al tratarse de un sistema con memoria virtual, no implica ningún acceso a las direcciones de la zona expandida y, por tanto, **ninguna invalidación**. Por otro lado, una disminución en el tamaño de una región es similar al caso analizado en el tercer punto (eliminación de una región): el sistema operativo tiene que **invalidar de la TLB las entradas de la región** que correspondan con páginas de la zona que han desaparecido al producirse la disminución.
6. Para duplicar eficientemente una región, se usa la técnica del *copy-on-write*, que retrasa todo lo posible el duplicado de cada página de la región hasta que se modifique su contenido. Para ello, cuando se pretende marcar como duplicada una región, se activa el bit de *copy-on-write* para las páginas de la región y se le quitan los permisos de escritura a cada página de la región para forzar un fallo cuando se intente modificar. Por tanto, dado que se cambian los permisos de todas las páginas de la región y esta información está incluida en las entradas de la TLB, el sistema operativo **debe invalidar las páginas de esa región** ya que la información sobre esas páginas en la TLB, si existe, ha dejado de ser coherente.
7. Si hay un fallo de página que encuentra un marco libre, dado que el marco que se usa está libre, eso implica que no puede existir ninguna entrada en la TLB asociada a ese marco (en un sistema convencional sin *buffering* de páginas, si antes estaba ocupado y ahora está libre, tiene que ser debido a que se ha invalidado o disminuido el tamaño de la región, como se analiza en los apartados respectivos, lo que conlleva su desaparición de la TLB por invalidación). Por tanto, no habrá **ninguna invalidación**.
8. En el supuesto de fallo de página que se plantea en el enunciado (no hay marcos libres, uso del algoritmo de reemplazo del reloj y que la primera página candidata a ser expulsada tenga tanto el bit de referencia como el de modificado a 1, mientras que la segunda tenga ambos bits a 0), el sistema operativo pondrá a cero el bit de referencia de la primera página candidata a ser expulsada, dándole una segunda oportunidad, y expulsaría la página contenida en el segundo marco usando éste como contenedor de la página que causó el fallo. La operación de poner a cero el bit de referencia requiere que el sistema operativo solicite la **invalidación de la TLB de la entrada correspondiente a la primera página** candidata a ser expulsada ya que, al cambiar el valor del bit de referencia, la entrada de la TLB, si existe, dejará de ser coherente. Con respecto a la página expulsada, el sistema operativo **no invalidará esta segunda página**, puesto que, como se acaba de analizar con la primera página, el no tener activo el bit de referencia indica que en la TLB no hay ninguna entrada vinculada con esta página.
9. Si en un fallo debido al *copy-on-write* se detecta que el contador de uso de la página es mayor que 1, hay que crear un duplicado de la página en un marco libre. En caso de que no hubiera marco libre, habría que buscarlo y se plantearía una situación similar al punto anterior. En cualquier caso, una vez que se tiene un marco libre, hay que proceder con el duplicado de la página, que es lo que se analiza en este punto. Como parte de esta operación, previamente el sistema operativo debe **invalidar la entrada** correspondiente a la página que causó el fallo de escritura, ya que, si está incluida en la TLB, hará referencia al marco que contiene la copia compartida.
10. Cuando sucede un fallo debido al *copy-on-write* con el contador de uso es igual a 1, implica que ya no hay más procesos que usan esta página. Por tanto, se le puede asignar al proceso que causa el fallo, permaneciendo correcta la traducción de la TLB asociada a esa entrada, en el caso de que exista. Sin embargo, el sistema operativo debe **invalidar la entrada** ya que hay que devolver el permiso de escritura a la página (este permiso se le quitó al marcarla como duplicada, como se analizó en el punto 6).

b1) En primer lugar, se analiza qué ocurre con la pila del sistema. Cuando un proceso cede el procesador durante el tratamiento de un evento síncrono del procesador, en la pila del sistema habrá almacenada información vinculada con la ejecución de dicha rutina (estarán guardados los registros de activación de las funciones que estén anidadas en ese momento).

Esta información debe mantenerse para, posteriormente, poder restaurar de forma transparente la ejecución de la rutina en el punto donde se quedó. Por tanto, es necesario que cada proceso use una pila de sistema diferente.
Por lo que se refiere a la pila de interrupción, la ejecución de una rutina de tratamiento de un dispositivo externo no puede provocar un cambio de contexto. Aunque una rutina de interrupción pueda verse, a su vez, interrumpida por otra de mayor prioridad, este anidamiento de rutinas se ejecutará en el contexto del mismo proceso (que, normalmente, no estará vinculado con las interrupciones), sin posibilidad de que haya cambios de contexto mientras tanto. Por tanto, se puede usar una única pila de interrupciones para todos los procesos.
Nótese que en las respuestas expuestas no influye si el kernel es expulsivo o no, como se analizará en los siguientes apartados.

b2) Un cambio de contexto voluntario se produce cuando el proceso en ejecución pasa al estado de bloqueado debido a que tiene que esperar por algún tipo de evento. Sólo pueden ocurrir dentro de una llamada al sistema o en el tratamiento de la excepción correspondiente al fallo de página. No pueden darse nunca en una rutina de interrupción, ya que ésta no está relacionada con el proceso que está actualmente ejecutando.
En el momento en el que se produce un cambio de contexto voluntario, la pila de usuario tendrá la información que corresponda con el anidamiento de funciones que había en el momento de producirse el evento. En el caso de una llamada al sistema, en la cima de la pila se encontrará el registro de activación correspondiente a la invocación de la "función de envoltura" de la llamada al sistema (la rutina de biblioteca que ofrece una interfaz funcional para realizar la llamada al sistema).
En cuanto a la pila de interrupciones, en el momento que se produce un cambio de contexto voluntario, estará vacía ya que sino se trataría la interrupción correspondiente antes de realizar el cambio de contexto.
El caso de máximo anidamiento se produce cuando, estando dentro del código de una llamada al sistema, se produce un fallo de página debido a que se intenta acceder a una página del mapa de usuario del proceso que no está residente (por ejemplo, para obtener un parámetro como el nombre del fichero que se quiere abrir). En esta situación, el estado de las pilas sería el siguiente:

- Pila de usuario. En su cima estará el registro de activación de la "función de envoltura" de la llamada al sistema.
- Pila de sistema. En la base habrá información correspondiente a la llamada al sistema. A continuación, estarán los registros de activación de las llamadas a función anidadas que estuvieran activas en el momento del fallo de página. Por encima de éstas, estará la información que introduce el procesador cuando se produce un fallo de página (al menos el contador de programa y el registro de estado correspondientes al momento en el que se produjo el fallo; en este caso dentro del código de una llamada al sistema) y el anidamiento de rutinas que haya invocado la rutina que maneja el fallo de página en el momento que realiza el cambio de contexto (si existe una función "bloquear" que encapsula la lógica del cambio de contexto voluntario, en la cima de la pila estará el registro de activación correspondiente a una llamada a esta función).
- Pila de interrupciones. Como se explicó previamente, estará vacía.

Por último, hay que hacer notar que el carácter expulsivo o no expulsivo del kernel no influirá al tratarse de un cambio de contexto voluntario.

b3) Un cambio de contexto involuntario se produce cuando el proceso en ejecución tiene que pasar al estado de listo ya que debe dejar el procesador por algún motivo (por ejemplo, debido a que se le ha acabado su rodaja de ejecución o porque hay un proceso más urgente que se ha desbloqueado por la ocurrencia de una llamada o una interrupción). Esta situación puede ocurrir dentro de una ejecución anidada de rutinas de interrupción. Para asegurar el buen funcionamiento del sistema, hay que diferir el cambio de contexto involuntario hasta que terminen todas las rutinas de interrupción anidadas, para lo cual se utiliza el mecanismo de interrupción software.
En este caso sí influye si el kernel es de tipo expulsivo o no:

- Si el kernel es expulsivo, el cambio de contexto se difiere sólo hasta que terminen las rutinas de interrupción. En el caso de que hubiera una llamada al sistema activa, su ejecución quedaría interrumpida en el punto donde estaba cuando se produjo el evento que causó el cambio de contexto involuntario. En este tipo de kernel, la interrupción software tiene más prioridad que la ejecución de una llamada al sistema.
- Si el kernel es no expulsivo, el cambio de contexto se difiere hasta que termine también la llamada al sistema. En este tipo de kernel, la interrupción software tiene menos prioridad que la ejecución de una llamada al sistema. Por tanto, no será tratada hasta que termine la llamada en curso.

Nótese que para ambos tipos de kernel el cambio de contexto involuntario se realiza siempre en la rutina de tratamiento de la interrupción software. La diferencia está en la prioridad de dicha interrupción con respecto a las llamadas al sistema.
En el momento en el que se produce un cambio de contexto involuntario, la pila de usuario tendrá la información que

corresponda con el anidamiento de funciones que había en el momento de producirse el evento. En el caso de que el cambio de contexto involuntario esté inducido por una llamada al sistema que desbloquea un proceso más prioritario que el que está actualmente en ejecución, en la cima de la pila se encontrará el registro de activación correspondiente a la invocación de la "función de envoltura" de la llamada al sistema (la rutina de biblioteca que ofrece una interfaz funcional para realizar la llamada al sistema).

Con respecto a la pila del sistema, en la cima de la misma estará la información asociada a la ocurrencia de una interrupción software (al menos el contador de programa y el registro de estado correspondientes al momento en el que se produjo esta interrupción) y el anidamiento de rutinas que haya invocado la rutina que maneja la interrupción software en el momento que realiza el cambio de contexto (si existe una función "cambio_contexto" que encapsula la lógica del cambio de contexto involuntario, en la cima de la pila estará el registro de activación correspondiente a una llamada a esta función). Si se trata de un kernel no expulsivo, está será la única información que hay en la pila del sistema, al tener la interrupción software menos prioridad que las llamadas. En el caso de un kernel expulsivo, la interrupción software puede haber interrumpido la ejecución de una llamada por lo que en la base de la pila del sistema estará esta información.

En cuanto a la pila de interrupciones, en el momento que se produce un cambio de contexto involuntario estará vacía ya que sino se trataría la interrupción correspondiente antes de realizar el cambio de contexto.

Para analizar el caso de máximo anidamiento y la influencia del carácter expulsivo del kernel, considérese una situación en la que estando ejecutándose el código de una llamada al sistema, se produce un fallo de página debido a que se intenta acceder a una página del mapa de usuario del proceso que no está residente (por ejemplo, para obtener un parámetro como el nombre del fichero que se quiere abrir) y en ese momento se produce una interrupción que desbloquea un proceso más prioritario activando una interrupción software.

En el caso de un kernel no expulsivo, terminaría la rutina de interrupción, el tratamiento del fallo de página y de la llamada al sistema y, justo en el momento de volver a modo usuario, entraría la interrupción software que llevaría a cabo el cambio de contexto involuntario. En esta situación, el estado de las pilas sería el siguiente:

- Pila de usuario. En su cima estará el registro de activación de la "función de envoltura" de la llamada al sistema.
- Pila de sistema. Tal como se explicó previamente, estará únicamente, la información correspondiente a la interrupción software.
- Pila de interrupciones. Como se explicó previamente, estará vacía.

Si se trata de un kernel expulsivo, también terminaría la interrupción, pero justo en ese momento entraría la interrupción software, en cuyo tratamiento se realizaría el cambio de contexto. Por tanto, el estado de las pilas sería el siguiente:

- Pila de usuario. En su cima estará el registro de activación de la "función de envoltura" de la llamada al sistema.
- Pila de sistema. Como se analizó previamente, la cima de la pila contendrá información correspondiente a la interrupción software, pero, debajo de la misma, se encontraría la información correspondiente a la rutina de fallo de página y, más abajo, a la llamada al sistema, que han sido temporalmente interrumpidas dado el carácter expulsivo del kernel.
- Pila de interrupciones. Como se explicó previamente, estará vacía.

b4) A continuación, se analiza cuál sería el tamaño máximo de cada pila y cuál es la influencia del carácter expulsivo del kernel en este análisis:

- Pila de usuario: Como se ha comprobado en los apartados anteriores, el tamaño de esta pila no está influido por el carácter expulsivo del kernel. Esta pila contendrá los registros de activación de las funciones que use el programa. Generalmente, los sistemas operativos modernos dejan que esta región crezca todo lo que haga falta, mientras que no se solape con la región más cercana. Sin embargo, dado que esta región requiere tener soporte en *swap*, numerosos sistemas operativos permiten que el administrador pueda fijar un límite máximo para el tamaño de esta región.
- Pila de interrupción: Esta pila tampoco está influida por el carácter expulsivo del kernel. Tiene que tener tamaño suficiente para guardar la información correspondiente al máximo anidamiento de interrupciones externas. Dado que las rutinas de interrupción suelen ser cortas y usar pocos recursos, generalmente es suficiente con asignar una página de la memoria del sistema operativo a esta pila, que, evidentemente, tendrá que estar siempre residente en memoria.
- Pila de sistema: Como se puede apreciar en el análisis previo, en el tamaño de esta pila sí influye que el kernel sea expulsivo o no.
    - En el caso de un kernel no expulsivo, el tamaño máximo se alcanza cuando hay un cambio de contexto voluntario pudiendo haber dos anidamientos correspondientes al tratamiento de una llamada y de un fallo de página dentro de ésta. En un cambio involuntario, sólo se incluye en la pila información vinculada con la interrupción software.
    - En el caso de un kernel expulsivo, el tamaño máximo se alcanza cuando hay un cambio de contexto

involuntario, puesto que, además de los dos anidamientos correspondientes al tratamiento de una llamada y de un fallo de página dentro de ésta, habrá un tercer nivel correspondiente a la interrupción software.

Dado que las rutinas del sistema operativo no tienen un uso de funciones con mucho anidamiento, generalmente es suficiente con asignar una página de la memoria del sistema operativo a la pila del sistema de cada proceso.