Diseño de sistemas operativos. Febrero de 2002. Ejercicio 1

Enunciado

Se pretende comparar distintas alternativas a la hora de acceder a un archivo, centrándose en qué activaciones del sistema operativo conlleva cada una de estas alternativas, en un sistema con las siguientes características:

- Memoria virtual basada en paginación por demanda (para el experimento, se considera que inicialmente todos los marcos están libres).
- Sistema de ficheros con un tamaño de bloque igual al tamaño de la página y que usa una cache con una política de gestión similar a la de UNIX (para el experimento, la cache se considera inicialmente vacía).
- Manejador de disco que sólo admite peticiones de 1 bloque.
- Algoritmo de planificación expulsivo basado en prioridades, utilizándose el mecanismo de interrupción software para realizar los cambios de contexto involuntarios.

En este sistema se están ejecutando sólo 2 procesos:

- Proceso P1, en el que se centra el experimento, que lee secuencialmente un fichero de N bloques.
- Proceso P2 de menor prioridad, que nunca se bloquea.

a) Suponiendo que P1 lee el fichero usando peticiones de lectura de 1 bloque, analice qué activaciones del sistema operativo conlleva completar una operación de lectura, especificando cuándo P1 pasa de modo usuario a sistema y viceversa, cuándo cambia de estado dicho proceso o en qué momento hay un cambio de contexto que le involucre, distinguiendo si es voluntario o involuntario. Asimismo, se debe explicar cuándo se producen transferencias de datos del fichero durante la operación de lectura, indicando quién las realiza (hardware o sistema operativo) y cuál es el origen y destino de la transferencia.

b) Repita el análisis del apartado anterior suponiendo que P1 usa la técnica de ficheros proyectados en memoria para leer el fichero. Explique razonadamente cuál de las dos soluciones resulta más eficiente.

c) Repita el primer apartado pero suponiendo que las lecturas son de 1 byte. Para ello, analice las dos primeras operaciones de lectura del fichero. Compare la eficiencia de esta estrategia con la usada en el primer apartado.

d) Repita el análisis del primer apartado pero suponiendo ahora que las lecturas son de 2 bloques. Compare la eficiencia de ambas alternativas. ¿Por qué cree que puede ser contraproducente usar peticiones de tamaño muy grande (por ejemplo, todo el fichero de golpe)? Para responder más concretamente a esta cuestión, considere una situación hipotética en la que el tamaño de la petición sea mayor que el tamaño máximo del conjunto residente establecido por una política de asignación de memoria fija. ¿Qué ocurriría con el buffer utilizado por el programa para leer?

e) Supóngase que P1 accede al fichero completo bloque a bloque, como en el primer apartado, pero para actualizarlo, sumando 1 a cada byte del fichero (o sea, lee un bloque, modifica su contenido y lo escribe, y así sucesivamente). Especifique cómo quedaría modificada la secuencia de activaciones planteada en el primer apartado al incluir la actualización, reflejando en qué momento se producen las escrituras al dispositivo.

f) Repita el apartado anterior suponiendo que P1 usa la técnica de ficheros proyectados en memoria para actualizar el fichero. Especifique cómo queda modificada la secuencia de activaciones planteada en el segundo apartado al incluir la actualización. ¿Ha encontrado nuevas diferencias en eficiencia entre estas dos técnicas (o sea, acceso convencional frente a proyección en memoria) al considerar las actualizaciones del fichero?

Solución

a) [1,5 puntos] Una operación de lectura de un bloque conlleva las siguientes activaciones del sistema operativo:

- Llamada al sistema solicitando la lectura de un bloque (llamada `read` en UNIX). P1 pasa de modo usuario a sistema. El procesamiento de esta llamada implica consultar el inodo del fichero, que se trajo a memoria en la apertura del mismo, para determinar qué bloque del dispositivo contiene el bloque solicitado, A continuación, se accede a la cache de bloques para averiguar si contiene dicho bloque. Puesto que inicialmente está vacía, no se encuentra el bloque y es preciso leerlo del dsipositivo. Antes de programar esta operación de lectura, es necesario buscar un bloque libre en la cache de bloques. Como inicialmente está vacía, los primeros accesos encontrarán un bloque libre (un poco más adelante, se comenta qué ocurriría si no fuera así). Una vez reservado el bloque, P1, ejecutando código del manejador del dispositivo, programa la operación de transferencia que usará la técnica del DMA, especificando como destino de la misma el bloque de la cache reservado. A continuación, P1 pasa al **estado de bloqueado** produciéndose un **cambio de contexto voluntario** a P2. A partir de ese momento, el **hardware irá transfiriendo por DMA al bloque de la cache** la información solicitada, mientras se está ejecutando P2.

  Es conveniente realizar dos aclaraciones sobre lo que se acaba de exponer:

  - Primero, hay que resaltar que si el bloque solicitado corresponde con un bloque indexado por punteros indirectos, habrá que leer previamente el bloque (o bloques, en el caso de varios niveles de indirección) de índices correspondiente. Este proceso de lectura previo presentaría las mismas características que el del propio bloque de datos (consulta de la cache, solicitud de lectura y bloqueo en caso de fallo, etc.).
  - Si a la hora de buscar un bloque libre en la cache todos están ocupados, será necesario reemplazar un bloque usado que, en caso de que estuviera modificado, habría que escribir previamente en el dispositivo, lo que implicaría el bloqueo correspondiente del proceso, como se comentará con más detalle en el penúltimo apartado de este ejercicio.
- Interrupción del dispositivo. Una vez finalizada la transferencia, el dispositivo genera una interrupción que activa la rutina de interrupción correspondiente que ejecutará en el contexto del proceso actual en ejecución (P2). Esta rutina, además de realizar las operaciones pertinentes sobre el dispositivo (como, por ejemplo, el control de errores), pasa a P1 de **bloqueado a listo** y, al detectar que tiene mayor prioridad, activa la interrupción software.
- Interrupción software. Realiza el **cambio de contexto involuntario** de P2 a P1. P1 continuará su ejecución en el punto donde se bloqueó, o sea, en el tratamiento de la llamada al sistema.
- Continuación de la llamada de lectura. El **sistema operativo realiza la copia desde el bloque de la cache hasta el buffer especificado en la llamada**. Además, hay que tener en cuenta que durante esta copia se pueden producir fallos de página si la página o páginas implicadas no están residentes (aunque el buffer tenga el tamaño de un bloque, pueden producirse hasta 2 fallos, ya que puede ocupar direcciones de 2 páginas sucesivas). El tratamiento de estos fallos de página no concierne directamente al ejercicio, que se centra específicamente en analizar el acceso al fichero. Cuando termina la transferencia, finaliza la llamada al sistema volviendo el proceso a modo usuario.

b) [1,5 puntos] Antes de analizar las invocaciones asociadas a la lectura del fichero, se repasa brevemente cómo se realiza la proyección de un fichero. Con esta técnica, una vez abierto el fichero de manera convencional, se invoca una llamada al sistema para proyectar el fichero (llamada `mmap` en UNIX). Durante esta llamada, el sistema operativo crea una nueva región vinculada el fichero, rellenando las entradas correspondientes de las tablas de páginas para uqe estén vinculada a los bloques del fichero. Es muy importante resaltar que durante esta proyección no se carga en memoria principal ningún bloque del fichero. Una vez proyectado, el proceso no tiene que usar ninguna llamada al sistema para leer o escribir en el fichero, ya que está directamente accesible en su mapa de memoria y puede usar instrucciones convencionales de acceso a memoria.

A continuación, se especifican las activaciones que conlleva el acceso usando esta técnica:

- Fallo de página. El primer acceso de P1 a una página de la región que corresponde con el fichero proyectado

causa un fallo de página que activa el sistema operativo. Como fruto de esta activación, el sistema operativo realiza el tratamiento típico de un fallo de página, buscando, en primer lugar, un marco libre. Como inicialmente, la memoria principal está vacía, los fallos asociados a los primeros accesos encontrarán marcos libres. Sin embargo, llegará un momento en el que esto no será así, y será necesario reemplazr una página usada que, en caso de que estuviera modificada, habría que escribir previamente en memoria secundaria,lo que implicaría el bloqueo correspondiente del proceso, como se comentará con más detalle en el último apartado de este ejercicio.

Una vez reservado el marco libre, P1, ejecutando código del manejador del dispositivo, programa la operación de transferencia que usará la técnica del DMA, especificando como destino de la misma el marco seleccionado. A continuación, P1 pasa al **estado de bloqueado** produciéndose un **cambio de contexto voluntario** a P2. A partir de ese momento, el **hardware irá transfiriendo por DMA al marco reservado** la página solicitada, mientras se está ejecutando P2.

- El tratamiento de la interrupción del dispositivo y de la posterior interrupción software será igual que en el primer apartado.
- Continuación del tratamiento del fallo de página. Se encargará de actualizar adecuadamente las estructuras del gestor de memoria. Al terminar esta rutina el proceso P1 vuelve a modo usuario.

Por lo que se refiere a la eficiencia, se puede apreciar que se producen el mismo número de activaciones del sistema operativo en ambos casos. Sin embargo, hay un ahorro muy importante al usar la proyección de ficheros: los datos se transfieren por DMA directamnete del dispositivo al proceso que los solicitó, eliminando la necesidad de pasar por la cache de bloques. Este ahorro es muy significativo ya que esa copia adicional por software consume bastantes recursos.

c) [1,5 puntos] Siguiendo la pauta especificada en el primer apartado, a continuación se analizan las activaciones asociandas a dos accesos consecutivos de un byte:

- Primera llamada de lectura. Su procesamiento es idéntico al del primer apartdao, ya que aunque se solicite un solo byte, siempre se trae completo a la cache el bloque implicado, Habría, por tanto, un cambio de contexto voluntario.
- El tratamiento de la interrupción del dispositivo y de la posterior interrupción software será igual que en el primer apartado.
- Continuación de la primera llamada. Es igual que en el primer apartado pero sólo se copia un byte de la cache al buffer de usuario.
- Segunda llamada de lectura. Al buscar en la cache, se encuentra el bloque que contiene el byte solicitado, que se copia directamente al buffer del usuario, retornándose directamente a modo usuario sin bloqueos.

Lo mismo ocurriría con las siguientes peticiones de lectura hasta que se consuma todo el bloque.

A la hora de leer secuencialmente el fichero, esta solución es apreciablemente menos eficiente ya que, aunque ocurre el mismo nú,mero de activaciones vinculadas a las interrupciones hardware y software y el mismo número de cambios de contexto, se producen muchas más llamadas al sistema con la consiguiente sobrecarga.

d) [2,5 puntos] (1,5 punto por análisis de activaciones para lecturas de 2 bloques + 1 punto por estudio de peticiones muy grandes)

Una petición de lectura de 2 bloques en este sistema implica las siguientes activaciones:

- Llamada de lectura. Dado que el manejador sólo admite peticiones de un bloque, el sistema operativo debe tratar cada bloque sucesivamente. Así, en primer lugar, el sistema operativo busca en la cache el primer bloque solicitado y, dado que la cache está inicialmente vacía, no lo encuentra. Se lleva a cabo el mismo tratamiento que en el primer apartado, programando la transferencia y realizando el cambio de contexto voluntario de P1 a P2.
- El tratamiento de la interrupción del dispositivo y de la posterior interrupción software será igual que en el primer apartado, provocando un cambio de contexto involuntario de P2 a P1.

- Continuación de la llamada. P1 realiza la copia del bloque como en el primer apartado y pasa a tratar el segundo bloque que, de la misma manera que el primero, causa un nuevo cambio de contexto voluntario de P1 a P2. Nótese que, durante esta última ejecución, P1 sólo ha estado en modo sistema.
- El tratamiento de la segunda interrupción del dispositivo y de la posterior interrupción software será igual que antes, provocando un cambio de contexto involuntario de P2 a P1.
- Continuación de la llamada. Realiza la copia del segundo bloque al buffer del usuario y termina volviendo P1 a modo usuario.

En este caso, aunque el número de interrupciones hardware y software es el mismo, se reduce a la mitad el número de llamadas al sistema, provocando uan ligera mejora en la eficiencia.

Sin embargo, según crece el tamaño de la petición, aunque se reduce el número de llamadas al sistema, se requiere tener en el mapa de memoria del proceso más información de forma simultánea. Esto puede implicar una pérdida de eficiencia, puesto que la memoria principal es un recurso limitado y parte de esta información puede ser expulsada a memoria secundaria.

Para ver más claramente este problema, sólo es necesario plantearse qué ocurriría en la situación planteada en el enunciado del ejercicio. Si una petición fuera más grande que el tamaño máximo del conjunto residente asignado al proceso, durante la operación de lectura, ñas sucesivas páginas que contienen el buffer se irían expulsando a memoria secundaria unas a otras. Cuando terminase la lectura, sólo estaría en memoria principal la parte final del buffer. Así, cuando el proceso comenzase a acceder a la parte inicial del buffer, e producirían fallos de página que, a su vez, expulsarían a las páginas finales del buffer. En resumen, la eficiencia se vería drásticamente afectada ya que habría que leer 2 veces la información: la primera vez se lee el bloque del fichero y la segunda se trae la página del dispositivo de memoria secundaria.

e) [1,5 puntos] Puesto que en UNIX se usa la técnica de *delayed-write*, una operación de escritura (llamada `write` en UNIX) sólo implica copiar los datos a la cahe de bloques. La escritura real al dispositivo se producirá sólo en dos situaciones: o bien cuando el bloque modificado fuera expulsado para traer otro o periódicamente (típicamnete cada 30 segundos).

La escritura, por tanto, sólo implicará una activación del sistema operativo:

- Llamada al sistema de escritura. Dado que se acaba de leer el bloque, todavía se encontrará en la cache una copia del mismo. Esta llamada sólo implicará copiar los nuevos datos desde el buffer del proceso (que estará probablemente residente en memoria principal y no dará fallo ya que se acabda de acceder al msimo) a este bloque de la cache, retornando a modo usuario sin haberse producido ningún cambio de contexto.

La lectura previa causará las activaciones que se detallaron en el primer apartado, aunque, dado que en este apartado se ha analizado un escenario donde se modificaba el fichero, se va a completar el estudio con una situación que quedo pendiente de detallar en el primer apartado: la secuencia adicional de activaciones que implica el caso de que no se encuentre un bloque libre y se tenga que expulsar un bloque modificado de la cache.

- Llamada al sistema de lectura. Si en el procesamiento de esta llamada explicado en el primer apartado no se encuentra un bloque libre, antes de tratar la lectura del bloque pedido, hay que seleccionar un bloque para reemplazarlo, que, si está modificado, habrá que escribirlo en el dispositivo. Para ello, se programaría la operación de escritura y se realizaría un cambio de contexto voluntario.
- El tratamiento de la interrupción del dispositivo y de la posterior interrupción software será el habitual.
- Continuación de la llamada. Se programa la lectura sobre el bloque recién liberado, se realiza un nuevo cambio de contexto voluntario.
- El resto de la operación es igual que en el primer apartado.

f) [1,5 puntos]

En este caso, la modificación del fichero no produce ninguna activación adicional del sistema operativo, ya que la página que se trajo cuando se trató el fallo de página asociado a la lectura, todavía está residente y se modificará sin

que intervenga el sistema operativo. La página modificada será escrita cuando sea expulsado durante el tratamiento de otro fallo de página.

Nótese que como ocurría en el caso anterior, durante el tratamiento del fallo de página estudiado en el segundo apartado, puede suceder que no haya marcos libres y que el seleccionada para ser expulsado esté modificado. La secuencia de activaciones planteadas en el segundo apartado, que quedó pendiente de detallar, se completaría de la misms forma que en el apartado anterior:

- Fallo de página. Al no encontrar un marco libre y el seleccionado para expulsar está modificado, antes de leer la página que causó el fallo, hay que escribir la página expulsada al dispositivo correspondiente. Para ello, se programaría la operación de escritura y se realizaría un cambio de contexto voluntario.
- El tratamiento de la interrupción del dispositivo y de la posterior interrupción software será el habitual.
- Continuación del tratamiento del fallo de página. Se programa la transferencia de la página requerida y se produce un cambio de contexto voluntario.
- El resto de la operación es igual que en el segundo apartado.

Por último, con respecto a la eficiencia, la mejora es notable ya que se eliminan las activaciones del sistema operativo vinculadas con la llamada al sistema de escritura. Nótese que `read+write` implica normalmente 4 activaciones del sisteam operativo (`read<`, interrupción del dispositivo, interrupcion software y `write`), mientras que los ficheros proyectados implican 3 (fallo de página, interrupción del dispositivo e interrupcion software).