Diseño de sistemas operativos. Septiembre de 2001. Ejercicio 1

Enunciado

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

a) En cada activación del S.O. se produce una de las siguientes alternativas:

- No cambia el estado de ningún proceso
- Cambia el estado de un proceso pero no hay cambio de contexto
- Hay un cambio de contexto voluntario
- Hay un cambio de contexto involuntario

Se pide que analice cuáles de esas 4 alternativas pueden ocurrir para cada uno de los 3 tipos de activación del S.O. que se detallan a continuación, planteando en cada caso una situación que sirva como ejemplo. Supóngase que en el sistema se usa un algoritmo de planificación expulsivo basado en prioridades.

1. Interrupción de reloj
2. Interrupción de disco
3. Llamadas al sistema

b) Dado un sistema de memoria virtual, se pretende analizar qué operaciones de lectura y escritura sobre el dispositivo de paginación requiere el tratamiento de un fallo de página.

1. Proponga un ejemplo de un fallo que no genere operaciones sobre el disco, otro que necesite sólo 1 lectura, otro que implique sólo 1 escritura y un último que requiera 1 lectura y 1 escritura.

2. Supóngase que se trata de un sistema que usa *buffering* de páginas. Analice cómo afecta esto a los ejemplos que ha planteado en el apartado anterior y proponga, en los casos que sea posible, ejemplos adicionales específicos de este tipo de sistema.

c) El diseño original de UNIX limita la concurrencia dentro del S.O. Esta estrategia tiene ventajas y desventajas.

1. En el lado positivo, facilita la programación del S.O. Proponga un ejemplo que avale esta afirmación.

2. En el lado negativo, limita drásticamente el uso de UNIX en entornos con características de tiempo real. En estos entornos es muy importante minimizar la latencia de activación de un proceso de alta prioridad (el tiempo que transcurre desde que un proceso de máxima prioridad pasa al estado de listo para ejecutar, hasta que realmente empieza a ejecutar). Explique por qué el diseño original de UNIX no favorece este requisito y analice cómo afectan, en caso de que así sea, a esta latencia de activación factores tales como el número de interrupciones existentes, el tiempo del tratamiento de cada interrupción y de cada llamada al sistema u otros factores que considere relevantes.

Solución

a) Se puede resaltar a priori que una interrupción nunca puede causar un cambio de contexto voluntario. Este tipo de cambios de contexto sólo pueden estar vinculados a una llamada al sistema que cause el bloqueo del proceso que la solicitó. Dado el carácter asíncrono de las interrupciones, la rutina de tratamiento de una interrupción ejecuta en el

entorno de un proceso que no tiene relación con la misma, por tanto, no debe causar el bloqueo de dicho proceso. A continuación, se presenta el análisis de cada una de las situaciones planteadas.

a1) Con respecto a la interrupción de reloj, excepto el cambio de contexto voluntario, pueden darse las otras 3 alternativas:

- La mayoría de las interrupciones de reloj no causarán el cambio de estado de ningún proceso. Simplemente, realizan las labores típicas asociadas a la interrupción de reloj (actualizan la hora del sistema, realizan labores de contabilidad, etc.).
- Una interrupción de reloj puede desbloquear a un proceso que solicitó previamente "dormir" un determinado plazo de tiempo. Si el proceso desbloqueado es menos prioritario que el actual, sólo hay un cambio de estado.
- Puede producirse un cambio de contexto involuntario en el caso de que, siguiendo la situación planteada en el punto anterior, el proceso que se desbloquea al cumplirse su plazo sea más prioritario que el actual.

Nótese que si el algoritmo de planificación utilizado fuera *round-robin*, también podría darse un cambio de contexto involuntario asociado a una interrupción de reloj que marcase el fin de una rodaja.

a2) El caso de la interrupción de disco es muy similar al de la interrupción de reloj. La principal diferencia es que las interrupciones de reloj ocurren sistemáticamente, sin que ningún proceso las tenga que "solicitar". Una interrupción de disco, sin embargo, está vinculada con una operación de disco que se inició previamente. Por tanto, no puede darse la situación de que no se produzca un cambio en el estado de algún proceso. Por lo demás, las alternativas son similares al caso anterior:

- Una interrupción de disco desbloquea al proceso que solicitó la operación. Si el proceso desbloqueado es menos prioritario que el actual, sólo hay un cambio de estado.
- Si el proceso desbloqueado es más prioritario, se produce un cambio de contexto involuntario.

a3) En cuanto a las llamadas al sistema, se pueden presentar todas las alternativas:

- Una llamada al sistema puede no producir ningún cambio de estado. Por ejemplo, una llamada que obtiene el identificador de un proceso nunca produce un cambio de estado. Tampoco lo hacen llamadas como una que lea datos de un terminal o una que "baje" un semáforo siempre que haya datos disponibles o el semáforo esté "abierto", respectivamente.
- Una llamada puede producir un cambio de estado pero no un cambio de contexto en el caso de que desbloquee a un proceso menos prioritario que el actual. Por ejemplo, una llamada que "suba" un semáforo o una que escriba en una tubería pueden causar el desbloqueo de un proceso (o sea, un cambio de estado bloqueado a listo) siempre que hubiera un proceso bloqueado en el semáforo o en la tubería, respectivamente. Si el proceso que se desbloquea es menos prioritario que el actual, sólo habrá cambio de estado.
- Puede producirse un cambio de contexto involuntario en el caso de que, siguiendo la situación planteada en el punto anterior, el proceso que se desbloquea sea más prioritario que el actual.
- Por último, también se puede producir un cambio de contexto voluntario si la llamada bloquea al proceso que la solicita. Retomando los ejemplos planteados en el primer punto, llamadas como una que lea datos de un terminal o una que "baje" un semáforo producirán un cambio de contexto voluntario si no hay datos disponibles o el semáforo está "cerrado", respectivamente.

b1) Antes de plantear los diversos ejemplos, es conveniente resaltar que, para que un fallo de página genere una operación de escritura, la rutina de tratamiento del fallo debe encontrarse con una situación en la que para traer la página que causó el fallo sea necesario expulsar una página modificada. A continuación, se muestran los ejemplos que corresponden con las distintas situaciones planteadas en el enunciado.

- **Sin operaciones**. Hay un fallo de página que corresponde con el primer acceso a una determinada página de una región sin soporte (como la región de datos sin valor inicial) y, o bien hay marcos libres, o la página elegida para reemplazo no está modificada. No hay lectura ya que sólo es necesario rellenar con ceros el marco elegido.
- **Sólo 1 lectura**. Hay un fallo de página que corresponde con una página que hay que leer del disco, ya sea porque pertenece a una región con soporte (como la región de datos con valor inicial) o porque está en el *swap*,

y, o bien hay marcos libres, o la página elegida para reemplazo no está modificada.
- **Sólo 1 escritura**. Hay un fallo de página que corresponde con el primer acceso a una determinada página de una región sin soporte (como la región de datos sin valor inicial), pero no hay marcos libres y la página seleccionada para el reemplazo está modificada. Hay sólo una escritura en disco del contenido de la página expulsada.
- **1 lectura y 1 escritura**. Hay un fallo de página que corresponde con una página que hay que leer del disco, ya sea porque pertenece a una región con soporte (como la región de datos con valor inicial) o porque está en el *swap*, pero no hay marcos libres y la página seleccionada para el reemplazo está modificada.

c) Antes de responder directamente a las preguntas planteadas, conviene recordar que el diseño original de UNIX, al igual que el de muchos otros sistemas operativos de propósito general, no permite que haya cambios de contexto involuntarios mientras se están procesando llamadas al sistema o interrupciones.

c1) Esta restricción facilita enormemente el control de las condiciones de carrera que se pueden dar entre llamadas al sistema ya que elimina la posibilidad de que éstas se ejecuten concurrentemente.

Así, por ejemplo, supóngase que el siguiente fragmento corresponde con el código de la llamada al sistema que crea un proceso:

```
crear_proceso(...) {
    ...........
    pos=BuscarBCPLibre();
    tabla_procesos[pos].libre=false;
    ...........
}
```

Sin esta limitación, podría ocurrir un cambio de contexto involuntario (por ejemplo, por fin de rodaja) justo después de que la función `BuscarBCPLibre` se haya ejecutado pero antes de poner la entrada correspondiente de la tabla de procesos como ocupada. El proceso activado por el cambio de contexto involuntario podría invocar también la llamada `crear_proceso` con lo que habría dos llamadas ejecutándose concurrentemente. Esta segunda invocación de `crear_proceso` causaría una condición de carrera ya que seleccionaría la misma posición de la tabla de procesos que la llamada interrumpida.

Gracias a la restricción de no realizarse cambios de contexto involuntarios mientras se están procesando llamadas al sistema, se eliminan este tipo de problemas facilitando la programación del sistema operativo.

c2) Debido a esta restricción, el cambio de contexto involuntario no se realiza justo en el momento en el que se produce el evento correspondiente, sino que su realización se retrasa hasta que termine todo el tratamiento de llamadas o interrupciones que esté en curso. Así, si una determinada rutina de interrupción desbloquea a un proceso de máxima prioridad, este proceso pasará a listo, pero el cambio de contexto en sí deberá esperar a que se ejecute todo el tratamiento de labores del sistema que pueda estar activo en ese momento. Téngase en cuenta que la interrupción "desbloqueadora" puede ser de máxima prioridad y, por tanto, pueden haberse anidado en ese momento distintas labores del sistema operativo:

- Un proceso de baja prioridad realiza una llamada al sistema.
- Justo al principio de la llamada, se produce una interrupción que detiene el tratamiento de la llamada y causa la activación de la rutina de interrupción correspondiente.
- Justo al principio de la rutina de interrupción correspondiente, se produce una interrupción de un nivel de prioridad superior.
- Así sucesivamente, hasta que llega la interrupción de máxima prioridad "desbloqueadora". En ese momento se desbloquea al proceso de máxima prioridad pero el cambio de contexto se diferirá hasta que terminen todas las rutinas anidadas.

Así, la latencia de activación del proceso de máxima prioridad queda afectada por la duración de las rutinas de tratamiento de las diversas interrupciones, así como por la duración de una llamada al sistema (en el peor de los casos, se vería afectado por la llamada al sistema más larga).

Nótese que, para ser precisos, esta latencia no estaría acotada ya que podrían ocurrir un número ilimitado de

interrupciones que harían que el cambio de contexto se aplazara indefinidamente, aunque, evidentemente, esto no es lo normal ya que generalmente el tiempo entre interrupciones de un mismo dispositivo es apreciablemente mayor que el tiempo de tratamiento de la interrupción.