Diseño de Sistemas Operativos. Febrero de 2009.
Ejercicio de procesos y memoria

Las operaciones de entrada/salida asíncronas, cuyo modo de trabajo consiste en iniciar la operación y devolver inmediatamente el control al programa, permiten aumentar la concurrencia de las aplicaciones sin necesidad de crear explícitamente flujos de ejecución concurrentes. Por ello, la mayoría de los sistemas operativos ofrece este tipo de operaciones tanto para el acceso a los ficheros como a los dispositivos de entrada/salida. La implementación de esta modalidad de entrada/salida presenta ciertas dificultades técnicas, algunas de las cuales se analizarán en este ejercicio. Para ello, se usará el siguiente esquema que describe de forma muy simplificada una hipotética implementación de una operación de lectura de fichero distinguiendo entre un modo síncrono y uno asíncrono. Evidentemente, en cada instante habrá múltiples peticiones de acceso al disco por parte de los distintos procesos, aunque uno de los aspectos que se ha obviado en este esquema básico es la sincronización entre las mismas.

```
leer_fichero() {
  Por cada bloque involucrado {
    Si está en caché
      Copia bloque de caché a zona usuario;
    Si no está en caché {
      Obtiene bloque libre en caché;
      leer_disco(petición);
      Si operación es síncrona {
        Bloquea_proc(cola asociada a bloq);
        Copia bloq de caché a zona usuario;
} } } }
leer_disco(petición) {
  Encola en lista de peticiones del disco;
  Si disco está libre
      Programa lectura a caché por DMA;
}
```

```
  int_disco() {
    Comprueba resultado de la operación;
    Activa int. software de sistema para
    tratar operaciones de interr. diferidas;
  }
  int_software_ops_disco_diferidas() {
    Desencola petición completada;
    Si peticiones pendientes en cola disco {
      seleccionar_próxima_petición();
      Programa lectura a caché por DMA;
    }
    Si operación completada es síncrona
      Desbloquea_proc(cola asociada a bloq);
    Si operación completada es asíncrona
      Copia bloque de caché a zona usuario;
  }
```

a) ¿Qué aspecto debe intentar optimizar la función `seleccionar_próxima_petición`? ¿Qué tipos de esquemas se usan en los sistemas operativos para llevar a cabo esta función?

b) ¿En qué acciones de la operación de lectura descrita en el esquema pueden producirse fallos de página?

c) [Especifique la traza de ejecución de la lectura síncrona planteada a continuación, identificando las activaciones del sistema operativo, los cambios de contexto, distinguiendo entre voluntarios e involuntarios, así como qué tipos de fallos de página se producen y cuándo se programa el disco. Considere un núcleo expulsivo, con *buffering* de páginas y tal que el tamaño de página sea igual al del bloque del sistema de ficheros.

- **P** (prioridad alta): solicita una lectura síncrona de dos bloques de un fichero que no están en la caché especificando como *buffer* una variable global sin valor inicial almacenada en dos páginas a las que no se había accedido previamente.
- **Q** (prioridad baja): Proceso recién creado que se dedica a realizar cálculos en modo usuario.

d) En el esquema planteado se ha optado por volver a programar el disco en el contexto de la interrupción, en caso de que haya peticiones pendientes, en lugar de hacerlo siempre, y exclusivamente, en el ámbito de la llamada al sistema (en concreto, en `leer_disco`). ¿Qué ventajas tiene la opción de diseño utilizada en cuanto a mejorar el grado de uso del disco? Para analizarlo, plantéese una situación con tres procesos de la misma prioridad en la cola de listos, tal que el primero (**A**) solicita leer síncronamente un bloque de un fichero que no está en la caché, el segundo (**B**) pide también una lectura síncrona de otro bloque que tampoco está en la caché, mientras que el último (**C**) no realiza ninguna operación vinculada con el disco. Nótese que no es necesario desarrollar la traza detallada de la ejecución, sino simplemente analizar para el ejemplo planteado el comportamiento de cada opción de diseño en cuanto al grado de utilización del dispositivo.

Centrándonos en la versión asíncrona del esquema planteado, ésta tiene un error de diseño, que es bastante frecuente a la hora de desarrollar manejadores de dispositivos.

e) Explique de forma razonada cuál es ese error de diseño, identificando qué acción no se está ejecutando en el contexto adecuado cuando se lleva a cabo una operación de lectura asíncrona.

f) Para solventar el error detectado, algunos sistemas operativos usan interrupciones software de proceso, de manera que la acción problemática identificada en el apartado anterior se ejecute en el contexto de una interrupción software de proceso (Windows usa una APC para tal fin). Explique qué cambios hay que hacer en el esquema original planteado para arreglar el error incorporando el uso de una interrupción software de proceso.

g) Basándose en esa solución, repita la traza del apartado **c**, pero cambiando los siguientes supuestos:

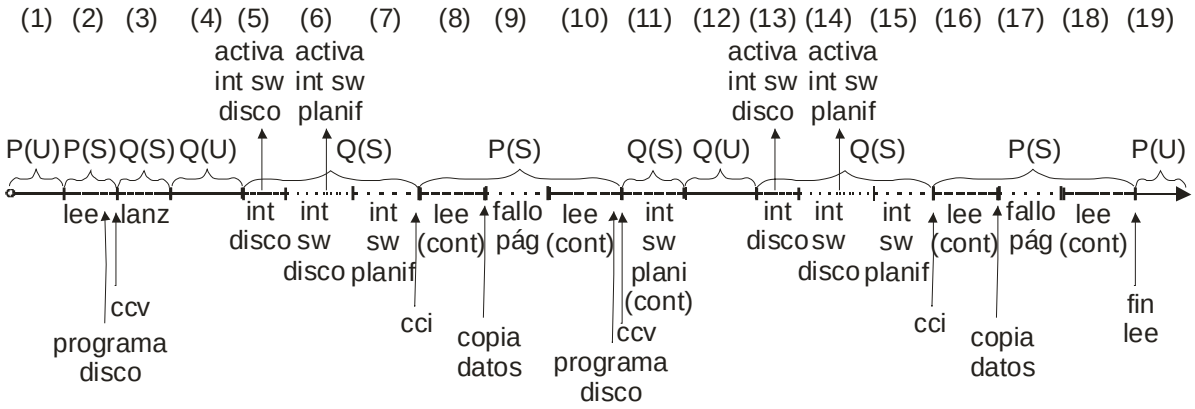
- **P** tiene prioridad baja y solicita la misma operación pero de forma asíncrona. Al retornar la llamada, realiza cálculos.
- **Q** tiene prioridad alta y está bloqueado esperando un plazo de tiempo que se cumple con la interrupción de reloj que se produce un cierto tiempo después de que se haya programado por primera vez el disco, pero antes de completarse esa primera operación. Al desbloquearse, ejecuta cálculos en modo usuario, interrumpidos por el fin de la primera operación del disco, y termina voluntariamente bastante antes de que se haya completado la segunda operación del disco.

Solución

a) La función `seleccionar_próxima_petición` implementa el algoritmo de planificación del disco, es decir, la función que selecciona en qué orden se sirven las peticiones de acceso al disco. El objetivo de esta función es optimizar el uso del dispositivo para lo que debe minimizar el movimiento de las cabezas del disco que se requiere para servir las peticiones. En la mayoría de los sistemas operativos se usan variantes del algoritmo del ascensor para implementar esta funcionalidad.

b) Dado que en la mayoría de los sistemas el mapa de memoria del sistema operativo está siempre residente, en el esquema planteado sólo se pueden producir fallos de página cuando se copia la información desde la caché del sistema al *buffer* especificado por el programa, que puede no estar completamente residente en memoria.

c) La siguiente figura muestra la traza de ejecución planteada en el enunciado:



A continuación, se detallan cada uno de los eventos que ocurren durante la traza:

1. **P** en modo usuario realiza una llamada al sistema solicitando leer de forma síncrona dos bloques de un fichero que no están en la caché, especificando como *buffer* una variable global sin valor inicial almacenada en dos páginas a las que no se había accedido previamente.
2. **P** en modo sistema comprueba que el primero de los bloques involucrados no está en la caché, busca un bloque libre en la misma y realiza la programación de la operación de lectura por DMA del disco, especificando como destino de la misma el bloque de la caché. A continuación, se bloquea en una cola de espera asociada al bloque pedido, produciéndose un cambio de contexto voluntario a **Q**.
3. **Q** comienza ejecutando en modo sistema una función de lanzadera que rápidamente se completa pasando a ejecutar en modo usuario la rutina inicial del programa.
4. Durante la ejecución en modo usuario de **Q**, se completa la operación de DMA produciéndose una interrupción del disco.
5. La rutina de interrupción del disco realiza unas labores básicas de carácter urgente, difiriendo las operaciones restantes mediante la activación de una interrupción software de sistema que está dispuesta para tal labor.
6. La rutina de interrupción software de sistema activada detecta que no hay peticiones pendientes, por lo que, dado que la operación completada es de tipo síncrono, simplemente se encarga de desbloquear al proceso que solicitó la operación. Dado que el proceso desbloqueado es más prioritario que el actual, se activa una interrupción software expulsiva de planificación para llevar a cabo el cambio de contexto involuntario de **Q** a **P**. Dado que el proceso interrumpido no estaba ejecutando previamente en modo sistema, la traza resultante sería igual para un núcleo no expulsivo.
7. La rutina de interrupción software de planificación lleva a cabo el cambio de contexto involuntario requerido, con lo que **P** continúa la ejecución de la llamada al sistema justo en el punto donde quedó detenida.
8. **P** en modo sistema comienza la copia de datos desde el bloque de la caché al *buffer* de usuario especificado en la llamada, pero, dado que no se había accedido previamente a la primera página que contiene el *buffer*, se produce un fallo de página que se anida con la llamada al sistema.
9. Al tratarse de una variable global sin valor inicial, estará incluida en una página de una región anónima, sin soporte, por lo que no será necesario leerla desde memoria secundaria. Además, al tratarse de un sistema con *buffering* de páginas, habrá marcos libres disponibles. En consecuencia, no habrá necesidad de acceder al disco para servir ese fallo de página, no requiriendo, por tanto, ningún cambio de contexto.
10. Al finalizar el tratamiento del fallo de página, se reanuda la llamada al sistema completándose el proceso de copia y pasando a tratar el segundo bloque involucrado. Dado que tampoco está en la caché, busca un bloque libre en la misma y realiza la programación de la operación de lectura por DMA del disco, especificando como destino de la misma el bloque de la caché seleccionado. A continuación, se bloquea en una cola de espera asociada al bloque pedido, produciéndose un cambio de contexto voluntario a **Q**.

11. **Q** reanuda su ejecución en el contexto de una rutina de tratamiento de una interrupción software de planificación donde quedó detenida su ejecución. Al completarla, retorna a modo usuario. A partir de este punto, se repiten los pasos desde el 4 hasta el 9 de la traza, que correspondían al acceso al primer bloque, por lo que sólo se hace una breve reseña de cada uno de ellos.
12. **Q** ejecuta en modo usuario hasta que se produce una interrupción del disco
13. Tratamiento de la interrupción del disco que activa la interrupción software de sistema que gestiona las operaciones diferidas.
14. Tratamiento de la interrupción software de sistema que desbloquea el proceso **P** y, al ser éste más prioritario, activa una interrupción software de planificación expulsiva para forzar el cambio de contexto involuntario.
15. Tratamiento de la interrupción software de planificación que realiza el cambio de contexto involuntario de **Q** a **P**.
16. Reanudación de la llamada al sistema que al comenzar el proceso de copia desde la caché a la segunda página que contiene el *buffer* de usuario provoca un fallo de página anidado.
17. Servicio del fallo de página, que no requiere operaciones de entrada/salida.
18. Cuando se finaliza el tratamiento del fallo de página, se completa la copia y con ello la llamada, puesto que no hay más bloques involucrados, retornando **P** a modo usuario.
19. **P** reanuda su ejecución en modo usuario justo después de la llamada de lectura síncrona.

Nótese que, según especifica el enunciado, el *buffer*, que tiene un tamaño suficiente para albergar dos bloques, está almacenado en dos páginas. Estará, por tanto, alineado con la dirección de comienzo de una página, por lo que se produce un fallo de página por cada bloque leído. En caso de no estar alineado, que sería lo más probable, estaría almacenado en tres páginas, pudiendo darse tres fallos de página: dos al copiar el primer bloque y uno al copiar el segundo.

d) Vamos a analizar el ejemplo de ejecución planteado para evaluar cuál de las dos opciones de diseño mejora el grado de utilización del disco. Usando el esquema especificado en el enunciado, se produciría la siguiente secuencia de ejecución:

- El proceso **A** solicita una lectura síncrona de un bloque que no está en la caché, encolando la petición y programando la operación por DMA del disco, para bloquearse a continuación.
- El proceso **B** también solicita una lectura síncrona de un bloque que no está en la caché. Sin embargo, en este caso, después de encolar la petición, al detectar que el disco está ocupado, se bloquea sin programar la operación por DMA del disco.
- Mientras el proceso **C** ejecuta sin realizar ninguna operación sobre el disco, se producirá una interrupción de disco que, a su vez, activará una interrupción software de sistema para tratar las operaciones diferidas. En el tratamiento de esta interrupción software de sistema se detectará que hay una petición pendiente (la de **B**) y se programará inmediatamente. Además, se desbloqueará el proceso **A**, pero como tiene la misma prioridad que el actual, éste continuará su ejecución. Obsérvese que se vuelve a programar el disco aunque ninguno de los procesos que están trabajando con el mismo estén en ejecución.

En esa secuencia de ejecución se puede comprobar cómo siempre que se completa una petición del disco y hay peticiones pendientes, se programa el disco para servir la petición seleccionada por el algoritmo de planificación del disco (función `seleccionar_próxima_petición`). Por tanto, el disco trabaja a pleno rendimiento.

Analicemos el caso de la segunda alternativa de diseño, es decir, que la programación del disco se produjera solamente en el contexto de la propia llamada al sistema de lectura. En este caso, dado que la programación del dispositivo la realiza el propio proceso que solicita la lectura, aunque el disco esté libre, la operación no se llevará a cabo hasta que le toque ejecutar al proceso involucrado. Así, en el ejemplo planteado, después de la interrupción del disco correspondiente a la operación del proceso **A**, que desbloqueará a dicho proceso, **C** prosigue su ejecución, quedando el disco inactivo aunque **B** tenga pendiente una petición de lectura: sólo cuando vuelva a ejecutar el proceso **B** se programará la siguiente operación.

e) En el esquema utilizado para la modalidad asíncrona de la operación de lectura de fichero se copian los datos desde la caché del sistema al *buffer* de usuario en el contexto de una interrupción software de sistema. Esa operación no es válida puesto que en ese contexto de ejecución no se conoce qué proceso está ejecutando en ese instante y, por tanto, no se sabe qué mapa de usuario está instalado, lo que puede hacer que se copien los datos en el mapa de un proceso que no solicitó la operación.

Esta operación de copia realizada en el ámbito de una interrupción software de sistema rompe dos reglas básicas sobre qué operaciones están permitidas en el contexto de una rutina asíncrona de sistema:

- Nunca se puede acceder al mapa de usuario en una rutina de estas características.
- Nunca puede haber bloqueos dentro de este tipo de rutinas. En el esquema planteado puede haberlos si durante la operación de copia se produce un fallo de página que requiere acceder al disco.

f) En el apartado anterior se ha identificado uno de los problemas de diseño que aparecen en la implementación de operaciones de entrada/salida asíncronas: ¿en qué contexto realizar el proceso de copia de los datos desde la memoria de sistema a la de usuario? Obsérvese que en el caso de las operaciones síncronas, este problema se resuelve haciendo que sea el propio proceso, una vez que se desbloquea, el que copie los datos en el contexto de la llamada al sistema de lectura. Sin embargo, esta idea no se puede aplicar a las operaciones asíncronas puesto que la llamada al sistema ya se completó y no podemos realizar ninguna suposición sobre en qué estado se encuentra el proceso involucrado en el momento en que se completa la operación.

La alternativa que plantea el enunciado, que usan algunos sistemas operativos, es realizar la operación de copia en el contexto de una interrupción software de proceso. Este tipo de interrupción software, al estar vinculada a un proceso específico, sí permite durante su tratamiento acceder al mapa de usuario activo o realizar un cambio de contexto, puesto que se conoce a priori qué proceso estará ejecutando en ese instante.

El esquema planteado debe modificarse en el caso de las operaciones asíncronas de manera que la copia de datos entre la caché del sistema y el *buffer* de usuario no se realice en el contexto de la interrupción software de sistema, sino en el ámbito de una interrupción software de proceso dirigida al proceso que solicitó la petición que se acaba de completar.

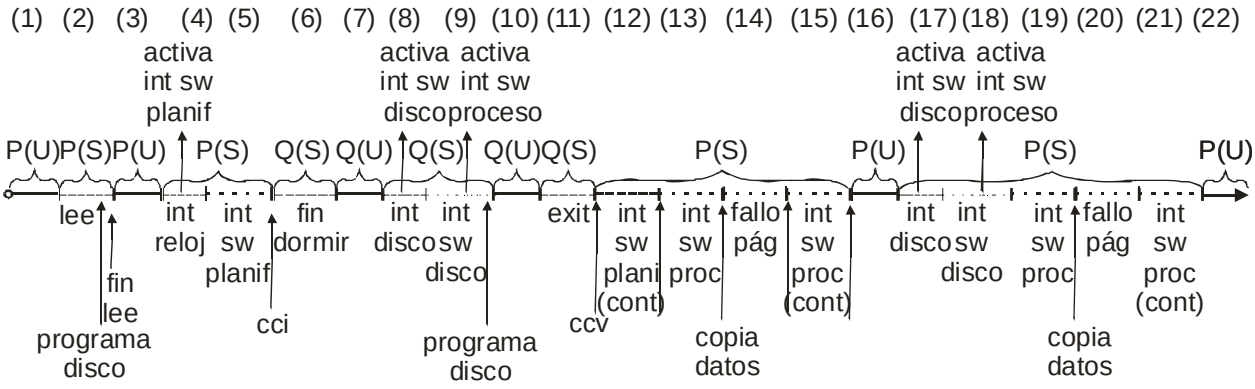
En el esquema resultante, en el tratamiento de la interrupción software de sistema se sustituye la copia de datos por la activación de una interrupción software de proceso asociada al proceso involucrado en la operación. Además, se añade el tratamiento de esta interrupción software de proceso donde se incluirá el proceso de copia (se ha subrayado el código añadido con respecto a la versión original).

```
leer_fichero() {
  Por cada bloque involucrado {
    Si está en caché
      Copia bloque de caché a zona usuario;
    Si no está en caché {
      Obtiene bloque libre en caché;
      leer_disco(petición);
      Si operación es síncrona {
        Bloquea_proc(cola asociada a bloq);
        Copia bloq de caché a zona usuario;
} } } }

leer_disco(petición) {
  Encola en lista de peticiones del disco;
  Si disco está libre
      Programa lectura a caché por DMA;
}
```

```
int_disco() {
  Comprueba resultado de la operación;
  Activa int. software de sistema para
  tratar operaciones de interr. diferidas;
}
int_software_ops_disco_diferidas() {
  Desencola petición completada;
  Si peticiones pendientes en cola disco {
    seleccionar_próxima_petición();
    Programa lectura a caché por DMA;
  }
  Si operación completada es síncrona
    Desbloquea_proc(cola asociada a bloq);
  Si operación completada es asíncrona
    Activa int.software de proceso dirigida
    a proceso que solicitó operación;
}
int_software_proceso() {
    Copia bloque de caché a zona usuario;
}
```

g) La siguiente figura muestra la traza de ejecución planteada en el enunciado:



A continuación, se detallan cada uno de los eventos que ocurren durante la traza:

1. **P** en modo usuario realiza una llamada al sistema solicitando leer de forma asíncrona dos bloques de un fichero que no están en la caché, especificando como *buffer* una variable global sin valor inicial almacenada en dos páginas a las que no se había accedido previamente.
2. **P** en modo sistema comprueba que el primero de los bloques involucrados no está en la caché, busca un bloque libre en la misma y realiza la programación de la operación de lectura por DMA del disco, especificando como destino de la misma el bloque de la caché. A continuación, trata el segundo bloque involucrado, que tampoco está en la caché, buscando un bloque libre y encolando la petición. En este segundo caso, al estar el disco ocupado, no se programa el disco, completándose el tratamiento del segundo bloque, con lo que concluye la llamada de lectura.
3. **P** en modo usuario ejecuta cálculos hasta que se produce la interrupción de reloj que esperaba **Q**.
4. La rutina de interrupción de reloj desbloquea a **Q**, y al detectar que es más prioritario que el proceso actual, activa una interrupción software de planificación para realizar un cambio de contexto involuntario diferido.
5. La rutina de interrupción software de planificación lleva a cabo el cambio de contexto involuntario de **P** a **Q**.
6. **Q** continúa su ejecución en modo sistema dentro de la llamada al sistema donde se bloqueó la última vez que ejecutó (*dormir*), completándola y retornando a modo usuario.

7. Durante la ejecución en modo usuario de **Q**, se completa la operación de DMA produciéndose una interrupción del disco.
8. La rutina de interrupción del disco realiza unas labores básicas de carácter urgente, difiriendo las operaciones restantes mediante la activación de una interrupción software de sistema que está dispuesta para tal labor.
9. La rutina de interrupción software de sistema, en primer lugar, detecta que hay una petición pendiente, por lo que programa una operación de acceso a disco para leer el segundo bloque. A continuación, al haberse completado una petición de acceso al disco que formaba parte de una operación asíncrona, activa una interrupción software de proceso dirigida al proceso que solicitó esa petición (**P**), que sólo se ejecutará cuando lo haga el proceso afectado por la misma. Cuando finaliza la rutina de interrupción software de sistema, **Q** retorna a modo usuario.
10. **Q** ejecuta en modo usuario hasta que invoca la llamada al sistema que indica que desea terminar su ejecución.
11. La llamada al sistema de finalización realiza un cambio de contexto voluntario de **Q** a **P**.
12. **P** reanuda su ejecución en el contexto de una rutina de tratamiento de una interrupción software de planificación donde quedó detenida su ejecución. Al completarla, antes de retornar a modo usuario, se activa el tratamiento de la interrupción software de proceso que tenía pendiente.
13. La rutina de interrupción software de proceso inicia la copia de datos desde el bloque de la caché a la primera página del *buffer* de usuario especificado en la llamada, pero, dado que no se había accedido previamente a la primera página que contiene el *buffer*, se produce un fallo de página anidado.
14. Servicio del fallo de página, que no requiere operaciones de entrada/salida.
15. Cuando se finaliza el tratamiento del fallo de página, se completa la copia y con ello la rutina de tratamiento de la interrupción software de proceso, retornando **P** a modo usuario.
16. **P** ejecuta en modo usuario hasta que se produce la segunda interrupción del disco
17. Tratamiento de la interrupción del disco que activa la interrupción software de sistema que gestiona las operaciones diferidas.
18. Tratamiento de la interrupción software de sistema que gestiona las operaciones diferidas que, al no haber peticiones pendientes, sólo activa una interrupción software de proceso dirigida al proceso que solicitó la petición que acaba de completarse (**P**), que en este caso coincide con el proceso actual, por lo que cuando finaliza la rutina de interrupción software de sistema, se activa inmediatamente el tratamiento de la interrupción software de proceso.
19. La rutina de interrupción software de proceso inicia la copia de datos desde el bloque de la caché a la segunda página del *buffer* de usuario especificado en la llamada, pero, dado que no se había accedido previamente a esa página, se produce un fallo de página anidado.
20. Servicio del fallo de página, que no requiere operaciones de entrada/salida.
21. Cuando se finaliza el tratamiento del fallo de página, se completa la copia y con ello la rutina de tratamiento de la interrupción software de proceso, retornando **P** a modo usuario.
22. **P** reanuda su ejecución en modo usuario.