DISEÑO DE SISTEMAS OPERATIVOS. Junio de 2004.

Ejercicio 1

Los sistemas operativos usan colas de espera para gestionar los procesos bloqueados esperando un evento. Responda razonadamente a las siguientes preguntas sobre este tipo de estructuras.

a) Analice qué colas de espera se requerirán en los siguientes operaciones del S.O., especificando a qué entidad estarán asociadas, y en qué activación del S.O. y bajo qué condiciones se encolarán y desencolarán los procesos.

1. En un manejador de disco.
2. En un manejador de terminal, distinguiendo si se trata de un terminal conectado a un puerto serie o una consola proyectada en memoria.
3. En la implementación de una tubería (*pipe*).
4. En el caso de que en un sistema monoprocesador con un único disco se produzca una secuencia de ejecución de procesos como la siguiente:
   - *P1* solicita escribir en un byte del primer bloque de un fichero, no estando ese bloque en la caché.
   - Justo después, sin terminarse la operación anterior, *P2* solicita leer otro byte del mismo bloque.
   - A continuación, sin finalizar ninguna operación previa, *P3* realiza por primera vez un acceso de lectura a una determinada página de la región de datos con valor inicial, habiendo marcos libres disponibles.
   - Después, sin finalizar las operaciones previas, *P4*, hijo de *P3*, accede en modo lectura a la misma página.

b) En algunos sistemas UNIX, se plantea un esquema alternativo de gestión de procesos en el que, manteniendo el carácter no expulsivo del sistema operativo, a un proceso que se bloquea en una cola de espera se le asigna una prioridad más alta que la de cualquier otro proceso del sistema que no esté en esta situación (el valor concreto de la prioridad depende del tipo de evento por el que se bloquea), con independencia de cuál sea la prioridad que tuviera previamente el proceso. Cuando continúa su ejecución después de desbloquearse, el proceso ejecuta con esta prioridad muy alta hasta que vuelve a modo usuario, momento en el que recupera su prioridad original.

1. ¿Cuál considera que es el motivo de usar esta estrategia?
2. Debido a su carácter no expulsivo, los SS.OO. UNIX no proporcionan buen soporte para procesos con características de tiempo real. A continuación, analice si con la nueva estrategia, aunque el S.O. siga siendo no expulsivo, mejoraría o empeoraría el tratamiento de los procesos con estas características.

c) Se plantea comparar el esquema de gestión de procesos convencional (o sea, en el que no se altera la prioridad del proceso cuando se bloquea) con el planteado en el apartado anterior. Supóngase un sistema monoprocesador con un único disco y un algoritmo de planificación de procesos basado sólo en prioridades. En este sistema un proceso *P1* de prioridad media solicita leer un bloque de un fichero que no está en la caché y, justo después, sin terminar la operación anterior, *P2* de prioridad baja ejecuta y causa un fallo en una página de código no habiendo marcos libres y estando modificado el marco que se pretende reemplazar. Analice cómo será la ejecución de estos procesos, especificando los cambios de contexto voluntarios e involuntarios que se produzcan, en los siguientes supuestos:

1. Se usa el esquema convencional y *P1* después de leer el bloque ejecuta un bucle de cálculo sin llamadas al sistema.
2. Se usa el esquema alternativo y *P1* después de leer el bloque ejecuta un bucle de cálculo sin llamadas al sistema.
3. Se usa el esquema convencional y *P1* después de leer el bloque ejecuta continuamente llamadas al sistema para obtener su identificador de proceso.
4. Se usa el esquema alternativo y *P1* después de leer el bloque ejecuta continuamente llamadas al sistema para obtener su identificador de proceso.

Solución

a) Antes de analizar cada uno de los casos planteados, hay que resaltar que el proceso de determinar qué colas se usarán para gestionar los procesos que están esperando por distintos eventos no tiene una única solución válida. En un extremo, se podría utilizar una única cola para todos los procesos bloqueados, sea cuál sea el evento por el que están esperando. Evidentemente, esta solución requeriría que se almacene para cada proceso bloqueado cuál es el motivo del bloqueo y conllevaría una gestión ineficiente, ya que el desencolado de procesos cuando se cumple el evento correspondiente implicaría recorrer la cola única de bloqueados buscando los procesos que estaban esperando ese evento. En el otro extremo estaría el dedicar una cola específica a cada evento individual, que proporciona una solución más eficiente, aunque requiere un poco más de espacio de almacenamiento para la información de control de las colas. En los sistemas reales, así como en esta solución, se usa esta segunda estrategia, aunque para algunos eventos relacionados, a veces se usa

una sola cola. A continuación, se analizan las operaciones planteadas en el enunciado:

1. Los discos son dispositivos de bloques. El manejador del disco ofrece operaciones para leer y escribir sobre los bloques del disco. Los usuarios de estas operaciones son, básicamente, el sistema de ficheros y el gestor de memoria.

   Con respecto al sistema de ficheros, se provocará una operación de lectura cuando un proceso realice una llamada al sistema para leer o escribir datos de un fichero que no están presentes en la caché de bloques (en el caso de la escritura, debe de tratarse de una operación que no involucre a bloques completos, ya que en ese caso se escribe completamente el contenido de los bloques por lo que no sería necesario leerlos del disco). Además, si para traer estos datos requeridos, es necesario expulsar algún bloque modificado, la propia operación de lectura o escritura del fichero requerirá una escritura previa en el disco del bloque expulsado. Además de escrituras debidas a expulsión, habrá escrituras periódicas al disco de los bloques modificados para asegurar que no se pierden datos. Estas escrituras son generalmente realizadas por un proceso del núcleo especializado.

   En cuanto al gestor de memoria, el tratamiento de los fallos de página del sistema de memoria virtual puede requerir una lectura del disco, ya sea del sistema de ficheros o del *swap*, y de manera similar al caso de los ficheros, pueden necesitar una escritura previa al dispositivo si es necesario expulsar una página modificada. En el caso de que se use la técnica del *buffering* de páginas, habrá escrituras cuando haya que expulsar páginas modificadas, debido a que no exista un número suficiente de marcos libres. Como ocurre con el sistema de ficheros, estas escrituras son habitualmente realizadas por un proceso del núcleo especializado.

   El manejador de disco mantiene una cola de las peticiones pendientes de satisfacer que hay para cada disco, ya sean generadas por el sistema de ficheros o por el gestor de memoria. A priori, puede parecer que bastaría con mantener una cola de procesos bloqueados en paralelo a la lista de peticiones, tal que cada proceso bloqueado en esta cola se corresponda con una petición en la lista de peticiones pendientes. Sin embargo, hay que tener en cuenta que esto no sería suficiente ya que puede ocurrir que un proceso necesite acceder a un bloque, ya sea por una llamada al sistema de ficheros o en el tratamiento de un fallo de página, que otro proceso ha solicitado previamente al manejador de disco pero tal que la petición todavía no se ha completado (podría haberse iniciado ya la operación o no). En este caso, no habría que hacer una nueva solicitud al disco, sino bloquear al segundo proceso en el mismo evento que al proceso previo. La solución habitual es que haya una cola de procesos bloqueados asociada a cada bloque solicitado (en el sistema de ficheros la cola puede estar incluida en la cabecera que describe cada bloque de la caché de bloques; en el gestor de memoria la cola puede estar en la información contenida en la tabla de marcos para cada página residente).

   Cuando un proceso solicita al manejador realizar una operación sobre un bloque de un disco y el dispositivo está libre, se programa la operación en el controlador del dispositivo y el proceso se bloquea en la cola de procesos asociada al bloque correspondiente. En el caso de que el disco se encuentre ocupado, se encola la petición y se bloquea al proceso en la cola asociada al bloque solicitado. Si mientras se sirve la petición de un bloque, otro proceso intenta acceder al mismo, el sistema de ficheros o el gestor de memoria, según el caso, detectan que el bloque ya ha sido pedido pero no se ha completado la operación, con lo que el proceso se bloquea en la cola asociada al bloque sin necesidad de contactar con el manejador del disco.

   En el momento que termina una operación en el disco, el controlador genera una interrupción cuya rutina de tratamiento se encarga, directamente o de forma diferida usando una interrupción software, de desbloquear a los procesos esperando por ese bloque. Además, si hay peticiones pendientes, selecciona una usando el algoritmo de planificación del disco y realiza la programación del controlador.

2. Además del tipo de terminal, hay que distinguir entre las operaciones de entrada y las de salida:
   - Operaciones de entrada en un terminal serie o en una consola. En este caso el tratamiento es similar para ambos tipos de terminales, ya que ambos están dirigidos por interrupciones, ya sea del puerto serie o del controlador de teclado, respectivamente. Por cada terminal, será necesario definir una cola de procesos bloqueados esperando que haya datos disponibles en el mismo. Téngase en cuenta que, en el caso de que el terminal esté en modo línea, sólo habrá datos disponibles en el momento que se complete una línea. Cuando un proceso realiza una llamada al sistema para leer del terminal y no hay datos disponibles en el *buffer* de entrada del terminal, se bloqueará en la cola de procesos asociados al mismo. Al llegar una interrupción del terminal que produzca datos disponibles para los procesos (nótese que si el terminal está en modo línea, no habrá datos disponibles hasta que llegue una interrupción correspondiente al carácter de fin de línea), si hay procesos bloqueados esperando datos, se desbloquearán.
   - Operaciones de salida en un terminal serie. En este tipo de terminales la salida va también dirigida por interrupciones. Cada vez que se solicita al puerto serie transmitir un dato, hay que esperar que llegue la interrupción para poder enviar el siguiente. Por tanto, es necesario tener una cola de procesos que quedarán bloqueados mientras se realiza la transmisión de un dato. Cuando un proceso realiza una llamada al sistema para escribir en un terminal de este tipo, se copian los datos en el *buffer* de salida del terminal y será necesario

bloquear al proceso en la cola mientras se transmiten.

- Operaciones de salida en una consola. En este caso sólo es necesario copiar en la memoria de vídeo la información correspondiente a la petición de escritura del proceso, no requiriéndose bloquear el proceso.

3. En el uso de una tubería, un proceso queda bloqueado cuando intenta escribir en ella estando llena o cuando intenta leer y está vacía habiendo descriptores de escritura para la tubería. Aunque puede parecer necesario usar dos colas de procesos por cada tubería, realmente sólo hace falta una ya que los estados de bloqueo son incompatibles: o bien hay uno o más procesos escritores bloqueados porque no hay sitio en la tubería o bien hay uno o más lectores bloqueados porque está vacía. El encolado se producirá en la llamada de lectura o escritura si se cumplen las condiciones anteriormente expresadas, y el desencolado (o desbloqueo) cuando se realice una llamada de escritura que genere datos (o el cierre del último descriptor de escritura sobre la tubería), para el caso de lectores esperando en una tubería vacía, o una llamada de lectura que consuma datos, para el caso de escritores bloqueados en una tubería llena.
4. La traza de ejecución sería la siguiente:
   - Como P1 intenta escribir sobre un byte (escritura incompleta) de un bloque que no está en la caché de bloques, solicita al manejador ese bloque. Al estar el disco libre, el manejador programa la operación y el proceso se bloquea en la cola de espera asociada al bloque solicitado.
   - Al comprobar si el bloque solicitado por P2 está en la caché de bloques, se detecta que el bloque ya ha sido solicitado previamente, por lo que P2 se bloquea en la cola de procesos asociada al bloque (la misma en la que está P1), sin contactar con el manejador.
   - En el tratamiento del fallo de página causado por P3 se comprueba que la página no está residente y se solicita el bloque correspondiente al manejador, que al comprobar que el disco está ocupado, encola una nueva petición y bloquea al proceso en la cola asociada al bloque pedido.
   - Cuando P4 provoca un fallo de página, se detecta en la tabla de marcos que la página requerida ya ha sido solicitada previamente, por lo que P4 se bloquea en la cola de procesos asociada al bloque (la misma en la que está P2), sin contactar con el manejador.
   - Cuando llega la primera interrupción del disco, se desbloquean los procesos P1 y P2 y se programa la siguiente petición.
   - Finalmente, llega la interrupción correspondiente a la segunda operación, que desbloquea a P3 y P4.

b) A continuación, se responde a las preguntas sobre el esquema alternativo de asignación de prioridad a los procesos.

1. Para comprender el beneficio de esta estrategia, hay que tener en cuenta que cuando un proceso se bloquea dentro de una llamada al sistema, puede haber reservado previamente distintos recursos del sistema operativo para asegurar la correcta semántica de la llamada, evitando que, mientras está bloqueado, otro proceso pueda usar estos recursos. Así, por ejemplo, para asegurar la atomicidad de las escrituras sobre un mismo fichero, teniendo en cuenta que cada escritura puede involucrar a varios bloques y, por tanto, generar varios bloqueos, habitualmente se usa un campo del inodo para marcar que está en uso. De esta forma, mientras un proceso que ha solicitado una escritura de múltiples bloques sobre un fichero está bloqueado esperando una transferencia del disco, si otro proceso invoca una operación de escritura en el mismo fichero, comprobará que está activa la marca y se bloqueará sin haber comenzado la operación. En esta situación, una vez desbloqueado el primer proceso, cuanto más tarde en volver a ejecutar, más tardará en liberar el inodo y, por tanto, en habilitar la ejecución del segundo proceso. Lo que plantea esta estrategia es dar una prioridad muy alta a los procesos que se han quedado bloqueados durante una llamada al sistema, para que, una vez desbloqueados, completen la llamada al sistema (o el fragmento de llamada hasta el siguiente bloqueo, si es una llamada con la posibilidad de que se produzcan múltiples bloqueos) lo antes posible liberando los recursos que pudieran tener retenidos, lo que redundará en un beneficio para los procesos que estaban esperando la liberación de dichos recursos.

   En resumen, se da prioridad a los procesos ejecutando el código del sistema operativo en modo núcleo, de manera que no pueda darse una situación en la que se esté ejecutando un proceso en modo usuario estando listo un proceso en modo núcleo. Nótese que, dado el carácter no expulsivo del núcleo de UNIX, mientras se está ejecutando una llamada al sistema que no bloquea al proceso, no puede entrar a ejecutar ningún otro proceso. Con esta nueva estrategia, se extiende esta propiedad no sólo al código de la llamada hasta el bloqueo, sino también al código de la misma posterior al bloqueo, en el caso de que lo haya.
2. Esta estrategia empeora el tratamiento de los procesos con un perfil de tiempo real, ya que con ella un proceso de tiempo real de alta prioridad puede ver interrumpida su ejecución por cualquier proceso, aunque sea de mínima prioridad, que se desbloquee, dado que en ese momento ese proceso desbloqueado adquiere una prioridad más alta hasta que complete la llamada al sistema que estaba realizando. Retomando lo comentado en el apartado anterior, con el nuevo esquema, los procesos no sólo pueden quedar retardados por el código de la llamada previo al bloqueo, si lo hubiera, sino por la parte final de la misma.

c) A continuación, se analizan las trazas de ejecución resultantes para los distintos casos planteados. Previamente, es conveniente recordar que el tratamiento de un fallo de página puede causar dos bloqueos al proceso en el caso de que haya que expulsar una página modificada, como ocurre en el ejemplo planteado, ejecutándose todo ello dentro de la misma activación del sistema operativo.

1. La traza de ejecución en este caso, que usa la estrategia convencional y en el que P1 realiza un bucle de cálculo, sería la siguiente:
   a. P1 solicita leer un bloque que no está en la caché de bloques. El manejador programa la operación y el proceso se bloquea en la cola de espera asociada al bloque solicitado, produciéndose un cambio de contexto voluntario a P2.
   b. P2 provoca un fallo de página. Se solicita el bloque correspondiente al manejador, que al comprobar que el disco está ocupado, encola una nueva petición y bloquea al proceso en la cola asociada al bloque pedido. Se produce un cambio de contexto voluntario al proceso nulo.
   c. Se produce una interrupción del disco que, en primer lugar, desbloquea a P1 que estaba en la cola de procesos asociada al bloque. A continuación, programa la operación de escritura de la página expulsada solicitada por P2. Por último, al detectar que el proceso desbloqueado es más prioritario que el actual, se activa una interrupción software para realizar un cambio de contexto involuntario diferido.
   d. Al finalizar la rutina de tratamiento de la interrupción del disco, se ejecuta la rutina de tratamiento de la interrupción software, que lleva a cabo el cambio de contexto involuntario del proceso nulo a P1.
   e. P1 termina la llamada de lectura del fichero y vuelve a modo usuario ejecutando un bucle de cálculo.
   f. Posteriormente, termina la operación de escritura produciéndose una interrupción del disco que, desbloquea a P2 que estaba en la cola de procesos asociada al bloque. Dado que P2 tiene menos prioridad que P1, no hay cambio de contexto, terminando la rutina de interrupción y retornando P1 a modo usuario.
   g. P1 termina su ejecución realizando un cambio de contexto voluntario a P2.
   h. P2 continúa con la rutina de tratamiento del fallo programando una lectura del disco y quedando bloqueado en la cola de espera asociada al bloque solicitado, produciéndose un cambio de contexto voluntario al proceso nulo.
   i. Se produce una interrupción del disco que, en primer lugar, desbloquea a P2 que estaba en la cola de procesos asociada al bloque. Al detectarse que el proceso desbloqueado es más prioritario que el actual, se activa una interrupción software para realizar un cambio de contexto involuntario diferido.
   j. Al finalizar la rutina de tratamiento de la interrupción del disco, se ejecuta la rutina de tratamiento de la interrupción software, que lleva a cabo el cambio de contexto involuntario del proceso nulo a P2.
   k. P2 termina la rutina de tratamiento del fallo de página y vuelve a modo usuario ejecutando la misma instrucción que causó el fallo.
2. La traza de ejecución en este caso, que usa la estrategia alternativa y en el que P1 realiza un bucle de cálculo, sería la misma hasta el paso e. A partir de ese punto, la traza seguiría de la siguiente manera:
   f. Posteriormente, termina la operación de escritura produciéndose una interrupción del disco que, desbloquea a P2 que estaba en la cola de procesos asociada al bloque. Dado que P2 tiene más prioridad que P1, ya que al bloquearse obtuvo una prioridad muy alta, se activa una interrupción software para realizar un cambio de contexto involuntario diferido.
   g. Al finalizar la rutina de tratamiento de la interrupción del disco, se ejecuta la rutina de tratamiento de la interrupción software, que lleva a cabo el cambio de contexto involuntario del proceso P1 a P2.
   h. P2 continúa con la rutina de tratamiento del fallo programando una lectura del disco y quedando bloqueado en la cola de espera asociada al bloque solicitado, produciéndose un cambio de contexto voluntario al proceso P1.
   i. P1 continúa con el bucle de cálculo hasta que se produce la interrupción del disco.
   j. Se produce una interrupción del disco que, en primer lugar, desbloquea a P2 que estaba en la cola de procesos asociada al bloque. Al detectarse que el proceso desbloqueado es más prioritario que el actual, se activa una interrupción software para realizar un cambio de contexto involuntario diferido.
   k. Al finalizar la rutina de tratamiento de la interrupción del disco, se ejecuta la rutina de tratamiento de la interrupción software, que lleva a cabo el cambio de contexto involuntario del proceso P1 a P2.
   l. P2 termina la rutina de tratamiento del fallo de página pero, cuando va a volver a modo usuario, se restaura su prioridad original, por lo que hay un cambio de contexto involuntario del proceso P2 a P1.
   m. P1 continúa su ejecución hasta que termina realizando un cambio de contexto voluntario a P2.
3. La traza de ejecución en este caso, que usa la estrategia convencional y en el que P1 realiza un bucle de llamadas al sistema no bloqueantes, será idéntica a la del primer caso. El único cambio que ocurre es que en el paso f, en el momento en el que llega la interrupción que indica el final de la operación de escritura solicitada por P2, el proceso P1 estará probablemente ejecutando una llamda al sistema, en vez de estar en modo usuario, como ocurría en el primer caso. Sin embargo, esta circunstancia no afecta a la traza de ejecución ya que el desbloqueo de P2 no causa un cambio de contexto pues tiene menos prioridad. Por tanto, no hay ningún cambio con respecto a la traza planteada en el primer

apartado: la rutina de interrupción terminará reanudándose la ejecución de la llamada interrumpida que se completará volviendo P1 a modo usuario y continuando su ejecución, un bucle de llamadas no bloqueantes, hasta que termine (paso g en la traza del primer apartado).

4. La traza de ejecución en este caso, que usa la estrategia alternativa y en el que P1 realiza un bucle de llamadas al sistema no bloqueantes, será prácticamente igual a la del segundo caso. La única diferencia se produce en los pasos f y j, que corresponden con el tratamiento de la interrupción del disco que indica que ha terminado una operación de P2 (en el paso f, se trata de la escritura de la página modificada, mientras que en el paso j, corresponde con la lectura de la nueva página). Esta diferencia consiste en que en este caso, como ocurría en el anterior, cuando llega esa interrupción, el proceso P1 estará probablemente ejecutando una llamda al sistema, por lo que la ejecución de la rutina de tratamiento de la interrupción software no sólo deberá esperar hasta que finalice la propia rutina de interrupción del disco, sino también hasta que termine la llamada al sistema de P1 interrumpida, dado el carácter no expulsivo del sistema operativo planteado.