5 Gestión de dispositivos

En este capítulo se presenta un modelo general de la gestión de los dispositivos de entrada/salida basado en un esquema cliente-servidor. Como ejemplo de gestión de un dispositivo en particular se estudian los discos, en razón de que son los dispositivos soporte del sistema de ficheros, prestando atención preferente a la evaluación del rendimiento.

Contenido

5.1 Introducción

Utilizaremos el término dispositivo para referirnos a cualquier elemento del computador que no sea el procesador o la memoria. Habitualmente los dispositivos se encargan de la entrada/salida, aunque la gestión de dispositivos hay que entenderla en sentido amplio, incluyendo los dispositivos de almacenamiento secundario y los de comunicaciones, e incluso la gestión del tiempo y de la energía. Esta heterogeneidad hace que el tratamiento de los dispositivos por el sistema operativo sea difícilmente generalizable para un estudio sistemático. En este capítulo vamos a intentar exponer un enfoque general de la entrada/salida, para después particularizar sobre un dispositivo concreto, centrándonos en los discos como dispositivo genérico para el soporte del sistema de ficheros.

5.1.1 Características de los dispositivos

Los dispositivos se caracterizan por su heterogeneidad, lo que introduce complejidad en el sistema operativo. Algunas de las características en las que los dispositivos pueden diferir son las siguientes:

- Unidad de transferencia. Unos dispositivos utilizan el byte como unidad de transferencia (**dispositivos de caracteres**, como el teclado o el ratón). Otros transfieren y/o almacenan la información en bloques (**dispositivos de bloques**, como discos y cintas magnéticas).

- Velocidad. Los rangos en los que se mueven los dispositivos son muy amplios. Los discos y los dispositivos de comunicación transfieren millones de caracteres por segundo y pueden hacerlo a velocidad constante, mientras que con el teclado se transfieren a lo sumo unos cuantos caracteres por segundo, con un periodo concreto impredecible.

- Representación de los datos. Incluso un mismo dispositivo puede utilizar diferentes codificaciones configurables en la instalación, como es el caso del teclado y el monitor.

- Protocolos de comunicación. La comunicación entre el dispositivo y la CPU se realiza de acuerdo a un determinado protocolo que depende del dispositivo y del bus de comunicación.

- Operaciones. Hay dispositivos de entrada, de salida y de entrada/salida. Además, algunos dispositivos requieren operaciones específicas (por ejemplo, posicionar el cabezal de lectura/escritura en los discos).

- Errores. Las condiciones de error varían con la naturaleza del dispositivo. Por ejemplo, en la impresora hay que tratar la falta de papel como una situación de error específica, mientras que en un disco puede haber errores en el posicionamiento del cabezal.

Para proporcionar una forma homogénea de direccionar los dispositivos, a nivel hardware éstos se conectan al sistema mediante **controladores**. El sistema operativo ya no trata con el dispositivo en sí mismo, sino con una interfaz que lo representa mediante un conjunto de direcciones o registros del controlador, que se pueden direccionar en el espacio de direcciones de memoria o constituir un espacio de direcciones independientes. El sistema se comunica con el controlador mediante operaciones de lectura/escritura sobre los registros de datos, estado y control, permitiendo tanto la transferencia de información como el diagnóstico y configuración del dispositivo. Estas operaciones las realizan las funciones de más bajo nivel del núcleo del sistema operativo, y son dependientes del hardware.

5.1.2 Tipos de entrada/salida

Por otra parte, y dependiendo en gran parte de las características del dispositivo, hay que distinguir tres tipos de entrada/salida, en función de cómo el sistema se sincroniza con el controlador:

E/S programada. La sincronización es por **encuesta**, realizándose un bucle de espera activa en la consulta del registro de estado del controlador. Los sistemas operativos multiprogramados evitan este tipo de operación.

E/S por interrupciones. El controlador activa una interrupción que permite la comunicación asíncrona del sistema operativo, que puede estar realizando otras tareas, con el dispositivo. Es la base que permite implementar un sistema operativo multiprogramado.

E/S por DMA. Los dispositivos de bloques, que requieren una tasa de transferencia muy elevada, utilizan el acceso directo a memoria para las operaciones de entrada/salida, bien utilizando ciclos de memoria libres (robo de ciclo), bien adueñándose de los buses de memoria para transferir un bloque completo. Este tipo de entrada/salida implica la utilización de interrupciones para la sincronización con el fin de la transferencia.

La evolución del hardware ha llevado a incluir capacidad de proceso dentro del dispositivo (**procesadores de E/S**). El sistema operativo se comunica con el procesador de E/S para indicarle los parámetros de la operación a realizar y ordenar su inicio. El procesador de E/S ejecuta un código propio que controla los detalles de la operación. Por otra parte, lo habitual hoy en día es incluir una cierta cantidad de memoria RAM en el controlador o en el dispositivo, sobre la que el sistema operativo

realiza la transferencia. Esto ocurre por ejemplo con los discos, las impresoras y las pantallas gráficas, que pueden contar con varios Mbytes de memoria.

Finalmente, un dispositivo puede estar accesible a través de una red, de forma transparente a las aplicaciones, situación habitual hoy en día, por ejemplo en las impresoras. Una pila de protocolos proporciona la comunicación entre la máquina cliente, que lanza la operación, y el servidor remoto, que gestiona el dispositivo. Este esquema cae fuera del ámbito de la asignatura objeto de estos apuntes.

5.2 Modelo general de la entrada/salida

Para facilitar su comprensión, el modelo de entrada/salida que vamos a presentar tiene como objetivo ser lo más sencillo y general posible, por encima de consideraciones acerca del rendimiento, e impone una estructura determinada al sistema operativo. Sus características fundamentales son:

- La entrada y salida se organiza y gestiona por capas, que responden a diferentes niveles de abstracción.
- El acceso a los recursos de entrada/salida se coordina de acuerdo al esquema cliente-servidor.

En el resto de esta sección estudiaremos ambas características como base del modelo general de entrada/salida. Los aspectos relacionados con la mejora del rendimiento pueden generalizarse en la utilización de *buffers* para el almacenamiento intermedio de la entrada/salida, que abordaremos en tercer lugar para completar el modelo.

5.2.1 Gestión de la entrada/salida por capas

Ante un panorama tan heterogéneo como el descrito en la introducción, abordar el estudio y diseño de la entrada/salida en un sistema operativo de manera comprensiva conduce a estructurar el sistema en diferentes niveles de abstracción o **capas**. Una capa L_k ofrece una **interfaz** a la capa superior, la capa L_{k+1}, conjunto de funciones que determinan la forma en que desde la capa L_{k+1} se accede a la capa L_k. En un sistema operativo la capa de más arriba corresponde a los programas de usuario, y está ya fuera de sistema operativo. Como ya se conoce, desde esta capa se accede a la capa inferior, ya dentro del sistema operativo, mediante la interfaz de llamadas al sistema. La interfaz de una capa ha de estar bien definida para que el programador sepa a qué atenerse. Por ejemplo, en los sistemas UNIX la interfaz de llamadas al sistema está definida en la sección 2 del *man*.

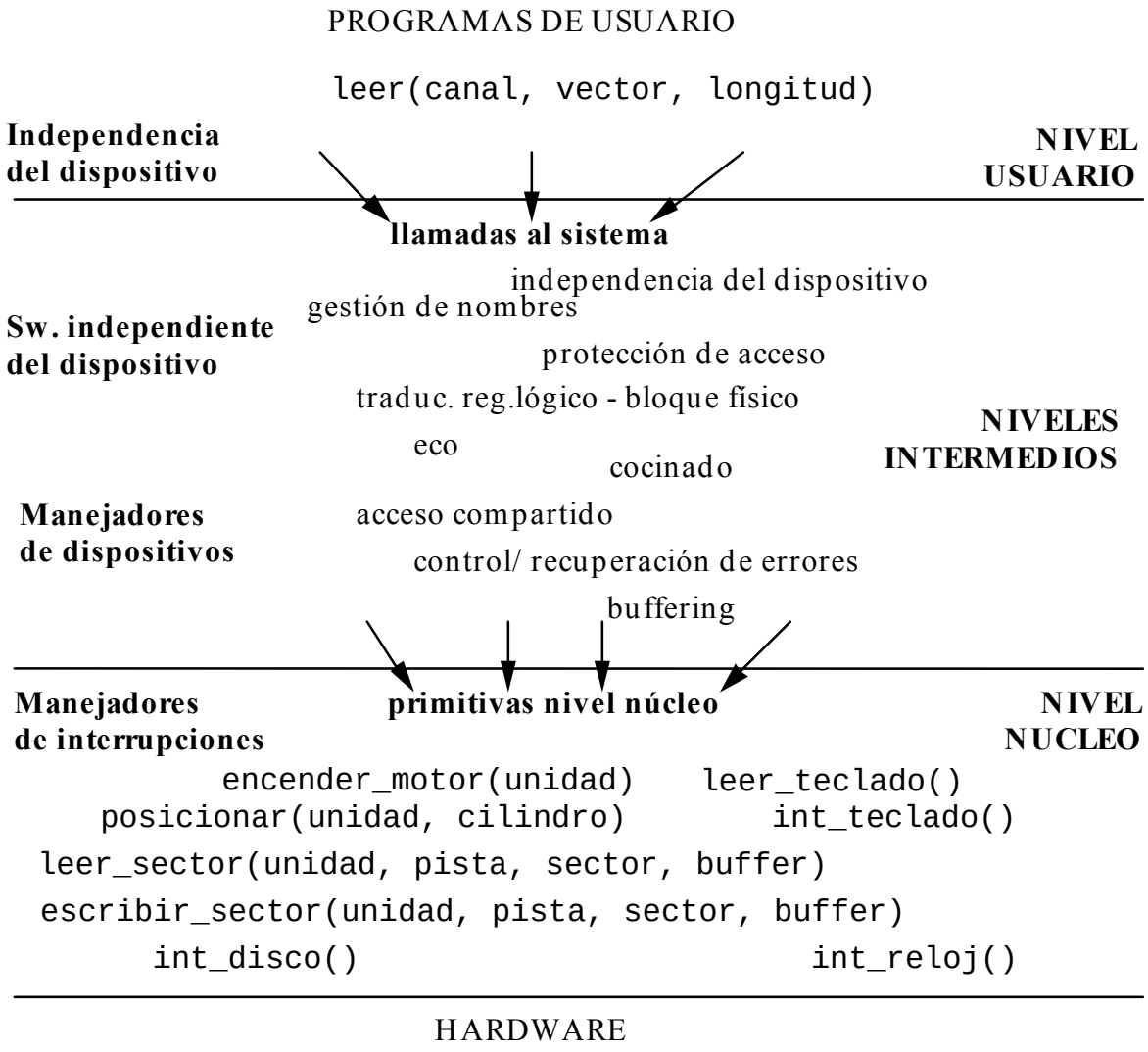

Figura 5.1. Gestión de dispositivos por capas

La estructura que vamos a proponer (Figura 5.1) consta de cuatro **capas** o niveles [TAN08][1]. De abajo arriba, el nivel más interno del sistema operativo (**núcleo**) programa los **controladores de los dispositivos** y maneja las **interrupciones**. Esta capa contiene software dependiente de los dispositivos y algunas partes han de ser codificadas en lenguaje máquina. Sobre el núcleo, en un segundo nivel se gestionan las peticiones de acceso a los dispositivos. Aquí residen los **manejadores de los dispositivos** (***drivers***), que tratan con las características particulares de los mismos y los controlan a través de las primitivas del núcleo. La tercera capa contiene **software independiente del dispositivo**: gestión de directorios, nombres, etc. Sobre estos niveles intermedios se monta la capa superior, que proporciona la **interfaz de**

[1] Este esquema en cuatro capas no es universal, sino el ejemplo de estructura que hemos elegido por su sencillez y claridad. Por otra parte, por razones prácticas, algunas capas pueden implementarse como una única.

llamadas al sistema para las aplicaciones y muestra los dispositivos como abstracciones que se representan por canales[2], proporcionando conceptos como el redireccionamiento de la entrada-salida.

5.2.2 Esquema cliente-servidor

Las operaciones de entrada/salida se especifican desde las aplicaciones mediante las llamadas al sistema, que trabajan con canales o dispositivos lógicos[3]. En general, una llamada al sistema típica (lectura o escritura) especifica de manera explícita o implícita los siguientes parámetros:

- La operación a realizar (leer, escribir…).
- El canal sobre el que se realiza la operación.
- La dirección (o posición) en el dispositivo E/S donde se accede. Normalmente está implícita (siguiente posición en un fichero) o incluso carece de sentido (lectura de teclado o ratón).
- La fuente o destino de la transferencia (dirección de memoria).
- La cantidad de información a transferir (longitud).
- En los sistemas que permiten operaciones síncronas y asíncronas, se indica esta condición y el evento con el que el programa que solicita la operación se va a sincronizar explícitamente.

El tratamiento de una operación de entrada/salida tiene dos partes. La primera, independiente del dispositivo, es el código utilizado por la llamada al sistema. Nos referiremos a ella como **rutina de E/S**. La segunda es el código del *driver* o **manejador del dispositivo**, y es dependiente del dispositivo. En nuestro modelo, la implementación del sistema operativo adopta el esquema cliente-servidor: las rutinas de E/S, ejecutadas por los procesos de usuario, corresponden a la parte del cliente del servicio, y el manejador, que se ejecuta como un proceso del sistema operativo, a la parte del **gestor** de la petición. A continuación presentaremos las estructuras de datos que proporcionan la interfaz entre las rutinas de E/S y los manejadores de los dispositivos, y que permiten hacer a la rutina de E/S independiente de las particularidades de la gestión del dispositivo.

---

2 En terminología Unix, descriptores de dispositivos.

3 En realidad, las aplicaciones utilizan las funciones de librería del lenguaje, que son las que utilizan las llamadas al sistema.

5.2.2.1 Representación de la E/S

La estructura que proporciona la comunicación entre la rutina de E/S y el manejador del dispositivo se suele denominar **IORB** (Bloque de Petición de E/S, *I/O Request Block*). La rutina de E/S utiliza un IORB para cada petición. Contiene la siguiente información:

- Identificación del proceso cliente.
- Parámetros de la petición.
- Evento para la sincronización del cliente con el final de la operación.
- Diagnóstico de la operación, a establecer por el manejador de acuerdo al resultado de la operación.

En un sistema operativo donde toda la E/S fuera síncrona, cada proceso dispondría de un IORB único y privado, asociado a su PCB, y el evento puede ir implícito. En un modelo general donde también es posible la E/S asíncrona, cada proceso puede disponer de IORBs de un conjunto, que reservaría en exclusión mutua, y el evento de sincronización sería explícito. En este modelo general, los procesos se bloquean por eventos en vez de por operaciones de E/S.

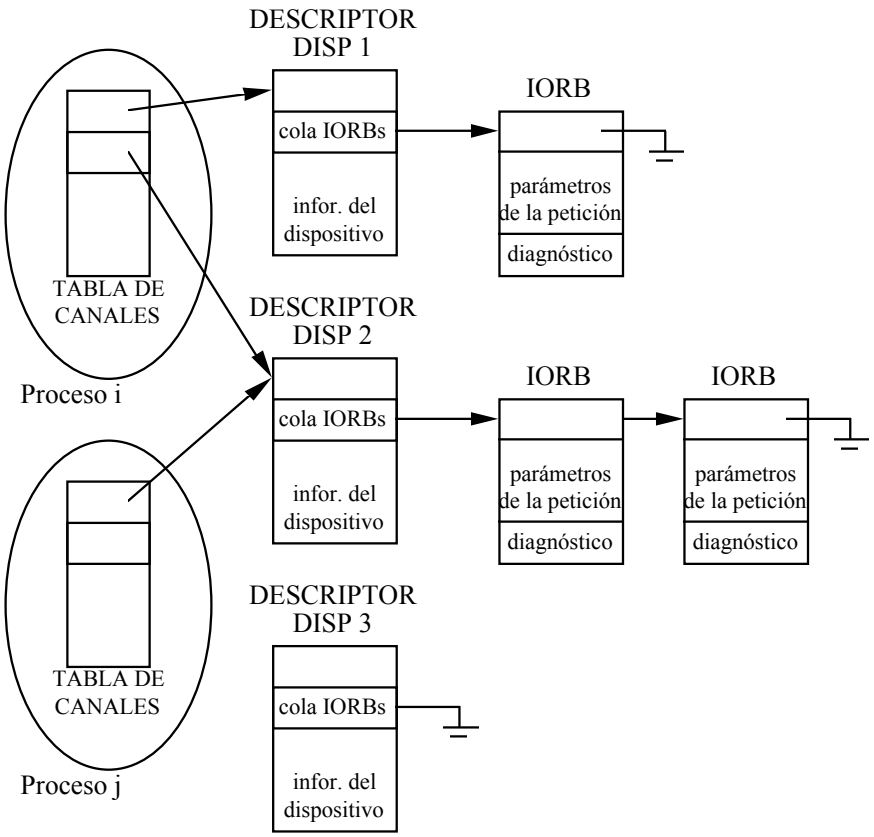

Figura 5.2. Un ejemplo del estado de la E/S en un sistema operativo

Las rutinas de E/S se mantienen independientes del dispositivo gracias a una estructura de datos asociada a cada dispositivo, el **descriptor del dispositivo**, que se direcciona a través de la tabla de canales y recoge las características del dispositivo y los parámetros propios de la operación con el dispositivo. Contiene información como:

- Estado del dispositivo.
- Modo de operación.
- Tablas de conversión.
- Apuntador a la cola de peticiones (IORBs) del dispositivo.
- Evento asociado al manejador correspondiente.

La Figura 5.2 representa un ejemplo del aspecto general de las estructuras de E/S en un sistema operativo de acuerdo al modelo introducido. Obsérvese cómo se integra en el modelo de colas de procesos descrito en el capítulo sobre la gestión de procesos.

5.2.2.2 Rutinas de E/S

Un proceso ejecuta una rutina de E/S poner una petición al manejador del dispositivo (driver). Gracias a que, como hemos visto, la tabla de canales del proceso especifica el descriptor del dispositivo correspondiente al driver, los detalles del manejador resultan transparentes a la petición, que tendrá el siguiente aspecto:

código_ret = petición_E/S (descriptor_del_dispositivo, operación, parámetros, evento_cliente)

El procedimiento que vamos a describir a continuación para ilustrar el funcionamiento de una rutina de E/S sigue un esquema que trata de ser general para todos los dispositivos (el acceso a ficheros requiere una etapa adicional de traducción de direcciones, como se verá en el capítulo correspondiente):

(1) A partir de la entrada de la tabla de canales, tras comprobar las características de la *operación* y los permisos de acceso, la rutina de E/S accede al *descriptor_del_dispositivo*.

(2) Construye la petición en un IORB libre con los *parámetros* de la petición.

(3) Encola el IORB creado en la cola de peticiones asociada con el dispositivo (apuntada desde el descriptor del dispositivo).

(4) Señala el evento de petición pendiente del manejador (especificado en el descriptor del dispositivo).

(5) Si la operación es síncrona, espera al *evento_cliente* asociado a la petición (en sistemas síncronos es implícito al proceso y no se especifica en la petición).

(6) Recoge el diagnóstico de la operación en el campo correspondiente del IORB. Interpreta las eventuales situaciones de error para devolver el código de retorno al proceso cliente.

(7) Libera el IORB.

El Ejercicio 6 ilustra un ejemplo de código de rutina de E/S, con las definiciones asociadas.

5.2.2.3 Manejadores de dispositivos

Son los manejadores asociados a los dispositivos. Un manejador contiene código dependiente de las características del dispositivo, por lo que el esquema de funcionamiento que se proporciona aquí es muy general. Para cada petición, el manejador o gestor del dispositivo está a la espera de que la rutina de E/S señale su evento, según se ha descrito más arriba, para atenderla. En un esquema cliente-servidor, el manejador realiza un bucle infinito, bloqueándose en cada iteración. A continuación se describe el esquema ejemplo de una iteración para el tratamiento de una entrada/salida por interrupciones típica en un dispositivo de caracteres.

(1) Cuando el manejador detecta un evento de petición pendiente, toma un elemento (IORB) de su cola de peticiones y extrae los parámetros de la petición.

(2) Programa la operación solicitada. Esto hace que el manejador se bloquee y que el núcleo del sistema operativo promueva un cambio de contexto.

(3) Espera por el final de la operación. Cuando se ejecute la rutina de atención correspondiente, el manejador se desbloquea y puede continuar.

(4) Transfiere la información a/desde el buffer especificado en el IORB. Para algunos dispositivos es preciso traducir la representación de los datos.

(5) Hace una comprobación de errores y escribe en el IORB el diagnóstico de la operación.

(6) Señala el evento especificado por el cliente en el IORB.

Si se trata de transferencias DMA (en dispositivos de bloques), el paso (4) va implícito y no lo gestiona el manejador, realizándose entre los pasos (2) y (3). Más adelante estudiaremos el esquema de un manejador de discos como ejemplo de dispositivo de bloques.

Por otra parte, las características propias de cada dispositivo determinarán la naturaleza de alguno de los pasos, fundamentalmente en lo que se refiere al tratamiento de errores y la traducción de la representación de los datos.

En nuestro modelo, presente en sistemas operativos con estructura cliente-servidor o basada en micronúcleo, el manejador del dispositivo es un proceso que se desbloquea para atender las peticiones. Este esquema resulta conceptualmente sencillo, pero no es muy eficiente, por lo que los sistemas operativos tradicionalmente han buscado implementaciones más directas basadas en sincronización mediante eventos de dormir/despertar o semáforos. En los sistemas UNIX, el proceso que quiere realizar una operación de entrada/salida se bloquea en un evento asociado al dispositivo hasta que este se libera, pasando a modo sistema para ejecutar el código del manejador y restaurando el modo anterior cuando finaliza la operación. De cualquier modo, el código del manejador ejecutado en este tipo de sistemas sigue los pasos descritos arriba para una iteración del servidor.

5.2.3 Almacenamiento intermedio de la E/S

Para desacoplar las velocidades de funcionamiento de los dispositivos con las de otros elementos del sistema y, por lo tanto, aumentar el rendimiento, es habitual la utilización de almacenamiento intermedio o ***buffering*** tanto de entrada como de salida.

El manejador del dispositivo proporciona **buffers del sistema** sobre los que se realiza la transferencia de entrada/salida. El buffer del sistema se copia a/desde el buffer de usuario, sobre el que la aplicación especifica la operación de E/S. El objetivo es doble. Por una parte, se amortigua la diferencia entre el ritmo de las peticiones de E/S y la capacidad del dispositivo para servirlas. Por otra parte, se consigue que, en sistemas con swapping y/o memoria virtual, las páginas de los programas de usuario que contienen los vectores especificados para la E/S no tengan que permanecer en memoria mientras el proceso está bloqueado[4]. Puede pensarse en diferentes esquemas de buffering, que dan lugar a diferentes formas de comportamiento de la entrada/salida en cuanto a la posibilidad de concurrencia entre las aplicaciones de usuario y los dispositivos (Figura 5.3):

(a) **E/S sin buffer**. La transferencia se realiza directamente sobre el buffer de usuario, que debe quedar fijado en memoria. Si una aplicación pretende transferencias asíncronas, debe gestionarse sus propios buffers.

[4] Este problema se planteó al hablar de la gestión de memoria.

(b) **Buffer simple**. No permite transferencias simultáneas, pero soporta transferencias asíncronas.

(c) **Buffer doble**. Introduce concurrencia al permitir simultáneamente dos transferencias (usuario-sistema en un buffer y sistema-dispositivo en otro). Este esquema se puede generalizar a N buffers.

(d) **Buffer circular**, que permite soportar concurrencia del tipo productor-consumidor. Un ejemplo característico es el buffer de anticipación del teclado.

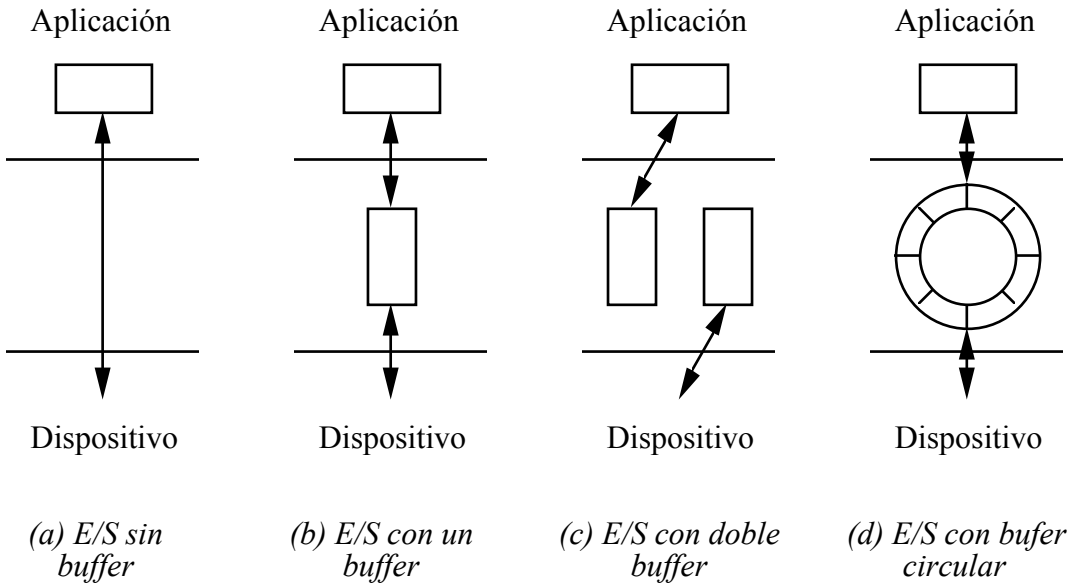

Figura 5.3. Esquemas de almacenamiento intermedio

La utilización de buffers mejora los parámetros de rendimiento temporal a cambio de espacio para implementarlos. También mejora la eficiencia de la CPU, aunque parcialmente compensada por el hecho de tener que copiar información entre los buffers del sistema y del usuario.

En algunos sistemas operativos la utilización o no de buffers se parametriza en la llamada al sistema de abrir y queda reflejada en la tabla de canales del proceso.

5.3 Gestión de discos

Estudiaremos en detalle la gestión del disco, dispositivo típico y común en los computadores de hoy en día, como ejemplo de las necesidades específicas que plantea al sistema operativo un dispositivo de cierta complejidad. Además los discos

constituyen tradicionalmente el soporte de los sistemas de ficheros[5], por lo que la gestión de discos se relaciona con la gestión de ficheros, que se estudiará en el siguiente capítulo. Aquí nos centraremos en los aspectos más relevantes de la gestión del disco, que se implementan fundamentalmente en el manejador del dispositivo.

Hoy en día, los discos pueden ser de diferentes tecnologías, magnéticos (rígidos o flexibles) u ópticos (de sólo lectura, como CD y DVD, o también de escritura, como CD-R, CD-WR y los diferentes formatos de DVD grabables). En lo fundamental, la gestión es similar para todos ellos, por lo que la mayor parte de los conceptos que vamos a introducir se aplican a cualquier tipo de disco. Salvo que se especifique lo contrario, tomaremos como referencia los discos magnéticos rígidos, que son el soporte más habitual para el sistema de ficheros. En general un disco se organiza en **sectores**, que se agrupan en **pistas**. Los discos rígidos pueden tener más de dos superficies (se agrupan en un paquete de discos que giran con el mismo eje), y entonces se habla de **cilindro** como el conjunto de pistas superpuestas en la misma vertical. Los discos flexibles, en vías de desaparecer, pueden tener una o dos superficies o **caras**. Para generalizar, aquí hablaremos de la jerarquía sector-pista-cilindro, aunque hay que advertir que lo habitual en los discos flexibles es utilizar la jerarquía equivalente sector-cara-pista; entendiéndose entonces por pista lo que antes hemos denominado cilindro.

Para que un disco sea utilizable en un sistema dado es necesario **dar formato** (o *formatear*), que incluye la inserción de información relativa al sistema de ficheros (como se verá en el siguiente capítulo), y para el control en los accesos a sectores.

El sector es la unidad de organización del disco visto éste como dispositivo físico. El sistema de ficheros agrupa los sectores en **bloques**, que es su unidad lógica de acceso y ubicación. Para la implementación del sistema de ficheros, el disco se ve como un vector de bloques[6].

---

[5] Los discos magnéticos, tras casi medio siglo de existencia, siguen siendo el soporte para el almacenamiento masivo en los sistemas de cómputo típicos de hoy en día (grandes servidores, ordenadores de sobremesa y portátiles). Además, se han introducido en otros dispositivos del mercado doméstico (por ejemplo, grabadores de video). Se trata de una tecnología compleja, que sigue madurando y avanzando, y que ha superado muchas hipotéticas barreras tecnológicas, consiguiendo gran fiabilidad y rendimiento a un coste muy reducido. Indudablemente, las memorias tipo *flash* (que se comercializan con el nombre de discos de estado sólido o unidades SSD) terminarán desplazando a los discos al menos cuando se requiera portabilidad, como ya ha sucedido en PDAs y en los ordenadores portátiles más ligeros. Aunque este cambio en el tipo de tecnología es radical, las consecuencias para las aplicaciones, e incluso para el propio sistema de ficheros, son inapreciables, gracias a que los sistemas operativos han adoptado esquemas que proporcionan independencia del dispositivo, como el modelo introducido en este capítulo.

[6] Los aspectos relacionados con la gestión de bloques se estudiarán en el siguiente capítulo.

El acceso a un sector del disco implica un conjunto de operaciones de posicionamiento: (1) movimiento de los cabezales hasta alcanzar el cilindro al que se accede, (2) selección del cabezal correspondiente a la pista, y (3) espera de rotación hasta alcanzar el comienzo del sector. Alguna de estas operaciones es de carácter mecánico, lo que introduce importantes retardos. Minimizar estos tiempos constituye un objetivo básico para la mejora el rendimiento. Una vez posicionado el sector correspondiente se lee/escribe de/en un buffer del controlador, donde se transfiere desde/hacia memoria principal. La inclusión de cierta cantidad de memoria RAM en los controladores modernos contribuye a minimizar el número de accesos al disco en sí, ya que los sectores accedidos pueden encontrarse en este buffer[7].

Dejaremos para la gestión de ficheros aspectos como la gestión de canales, la protección de accesos, la gestión de directorios y la ubicación de bloques (que se pueden resumir en la traducción de registro lógico de un fichero a número de bloque donde reside en disco), y que constituye la parte específica de la rutina de E/S en el tratamiento de ficheros. Aquí vamos a considerar estrictamente las funciones propias de la gestión del dispositivo disco:

- Arranque/parada automática del motor (en discos flexibles y ópticos).
- Traducción de número de bloque a cilindro, pista y sector.
- Planificación de accesos (gestión de peticiones).
- Control/recuperación de errores (con reintento).

Prestaremos especial atención a los aspectos relacionados con la evaluación del rendimiento y, finalmente, particularizaremos el esquema general cliente-servidor al manejo de los discos.

5.3.1 Arranque y parada del motor

Los discos rígidos giran continuamente y el motor sólo se para excepcionalmente (si se configura para ello), ya que los accesos suelen ser frecuentes. La mecánica de las unidades de discos rígidos es extremadamente precisa, no hay rozamientos[8] y disipan poco calor. En cambio, los discos flexibles y los ópticos no pueden estar

---

[7] Nótese que este buffer reside en el propio dispositivo y es independiente de los buffers del sistema operativo, que residen en memoria principal. Ambos buffers no se excluyen. Como se verá más adelante, el sistema de ficheros se encarga de gestionar los buffers de memoria principal para minimizar el número de peticiones de acceso a bloques al manejador, mientras que la gestión del buffer del dispositivo la realiza el controlador en los accesos a sectores solicitados por el manejador.

[8] Los discos están encapsulados en una caja donde se ha hecho el vacío.

girando continuamente. Como además los accesos a estos últimos suelen ser puntuales y menos frecuentes, lo normal es que el manejador encienda el motor cuando va a realizar un acceso. El apagado puede programarse tras un tiempo sin usar, lo que involucra a la rutina de atención al reloj.

5.3.2 Traducción de bloques

El manejador del disco recibe peticiones de acceso a bloques por parte de los procesos. Dado un número de bloque en el disco, *b*, éste se traduce a número de cilindro, *c*, número de pista en el cilindro, *p*, y numero de sector en la pista, *s*. Si el sistema operativo maneja bloques de *N* sectores, la operación involucra el acceso a *N* sectores consecutivos, y los parámetros del disco obtenidos se refieren al primero de los sectores de ese bloque[9].

Dado un disco con una geometría de *C* cilindros de *P* pistas de *S* sectores (particularizado en el ejemplo de la Figura 5.4 con valores (C, P, S) = (3, 2, 4)) se deducen las fórmulas que permiten obtener los parámetros del acceso en función del número de bloque. El disco contará con $C \cdot P \cdot S$ sectores y el número de bloques total, *B*, será un divisor entero de esta cantidad: $B=CPS/N$. Si "/" representa la división entera y "%" la operación módulo:

$$c = bN / PS$$

$$p = (bN / S) \% P$$

$$s = bN \% S$$

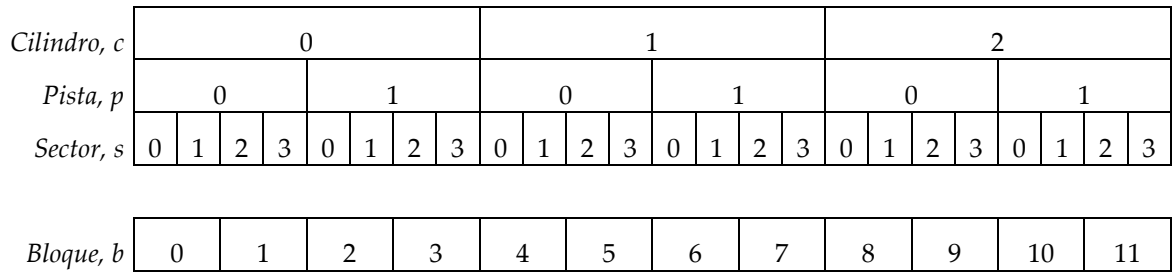

Figura 5.4. Representación de un disco con geometría (C, P, S) = (3, 2, 4). El tamaño de bloque, N, es de 2 sectores.

[9] Los discos rígidos de hoy en día realizan este proceso de traducción por hardware. Además, los dispositivos suelen entender el parámetro absoluto de acceso a un sector como una entidad lógica, lo que les otorga la capacidad de reubicar sectores dañados en otras zonas del disco.

5.3.3 Reubicación de sectores. Interleaving de sectores.

En muchos casos, el valor de número de sector *s* obtenido de acuerdo a lo explicado anteriormente puede no corresponder al s-ésimo sector físico de la pista, sino referirse a un número de sector *lógico* que requiere aún una ulterior etapa en la traducción. Históricamente esto se justifica por lo siguiente. Un acceso al disco supone una transferencia DMA entre el dispositivo y la memoria. Supongamos una operación de lectura. Si el buffer del controlador del disco es de un tamaño tal que sólo puede contener un sector, el acceso al siguiente sector de la pista no podrá comenzar hasta que el buffer se haya transferido completo a memoria. Pero, durante la transferencia, el comienzo del siguiente sector habrá pasado debajo del cabezal de la unidad sin poder ser leído o escrito, y habrá que esperar a que se posicione en la siguiente rotación para accederlo.

Por este motivo, algunos controladores numeran los sectores de manera no correlativa en la pista, para que, accedido el sector *i*, el tiempo de posicionamiento en el *i+1* se minimice. Esta técnica se conoce como ***interleaving* de sectores**. De acuerdo al tiempo de retardo producido por la transferencia de un sector, entre los sectores *i* e *i+1* se intercala uno o más sectores, de forma que el tiempo de rotación para saltar estos bloques sea ligeramente superior al retardo de la transferencia.

Se denomina **factor de interleaving** al número de sectores que se intercalan entre dos sectores lógicamente consecutivos. Cabe hablar entonces de **sectores lógicos** y **sectores físicos**. La Figura 5.5(a) muestra la ordenación física de los sectores lógicos en una pista sin interleaving. La Figura 5.5(b) muestra cómo se ordenarían físicamente con un interleaving de factor 2. El factor de interleaving elegido será óptimo si minimiza el tiempo de posicionamiento en el siguiente sector.

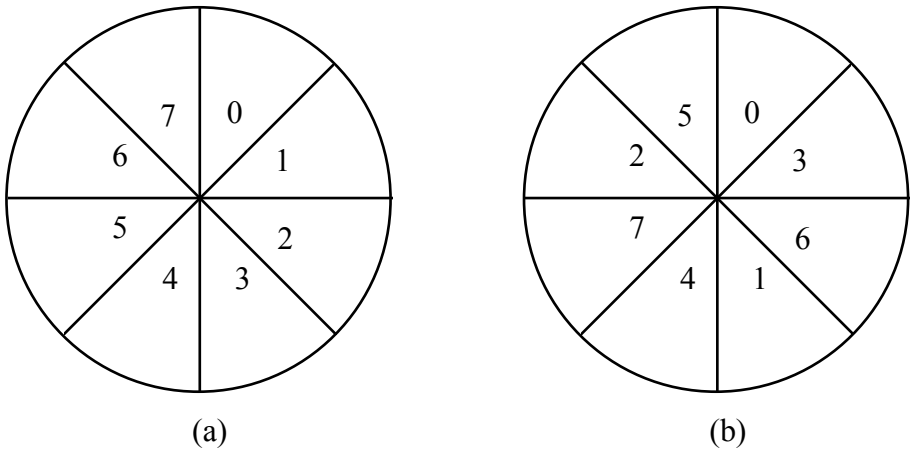

Figura 5.5. Ordenación de sectores lógicos: (a) sin interleaving, (b) con interleaving factor 2.

La forma de implementar el interleaving es mediante una tabla que proporciona la traducción de sectores lógicos a sectores físicos. Cuando es el controlador quien gestiona la traducción de sectores, se habla de **interleaving hardware**. Si el controlador no proporciona interleaving, el propio sistema operativo puede gestionarlo. En este caso se habla de **interleaving software**.

Hoy en día los controladores disponen de buffers de gran tamaño que permiten almacenar pistas enteras, por lo que la utilización del interleaving de sectores pierde su sentido. Sin embargo, la distinción entre sectores físicos y lógicos sigue teniendo interés, como veremos, para la gestión de sectores dañados.

5.3.4 Planificación de accesos

El acceso a un bloque en disco consta de varias operaciones con retardos relacionados con las características físicas del dispositivo. Mientras que la selección de cabezal es una operación de carácter eléctrico y, por lo tanto, presenta un retardo nulo, el posicionamiento en un cilindro y la espera de rotación son operaciones de carácter mecánico cuyo retardo es importante y debe minimizarse para mejorar el rendimiento del sistema. Este es uno de los objetivos del manejador del disco. Globalmente, la estrategia afecta al tiempo de rotación y de movimiento del cabezal. La espera de rotación se reduce con el interleaving de sectores o utilizando buffers del tamaño adecuado. El posicionamiento en cilindro es, en un sistema multiprogramado, un problema de ordenación de las peticiones.

Ordenando las peticiones se puede minimizar el tiempo medio de posicionamiento del cabezal en el comienzo de un bloque. Aunque el orden afecta también al tiempo de rotación, aquí vamos a considerar únicamente estrategias basadas en el número de cilindro. Combinando ambas operaciones pueden concebirse políticas globales más complejas.

Supondremos que el manejador de disco, en un instante de tiempo dado, dispone de una cola de peticiones de acceso a disco. Cada petición se especifica como el número de cilindro al que se quiere acceder. Se trata de reordenar las peticiones de la cola de forma que la distancia recorrida por el cabezal sea la menor. Aunque el tiempo de traslación del cabezal no es lineal con la distancia, consideraremos esta aproximación suficiente. Tenemos las siguientes políticas de gestión:

FCFS* o *FIFO

No requiere ningún tipo de proceso sobre la cola de peticiones. Es la más sencilla pero poco eficiente.

***SSF* (el más cercano primero)**

No garantiza que el cabezal recorra la distancia mínima. Su principal problema deriva del hecho de que los cilindros centrales son los más cercanos en media a los demás cilindros. Como la cola de peticiones es dinámica, esto puede producir inanición en los cilindros extremos.

***SCAN o LOOK* (algoritmos del ascensor)**

El cabezal recorre los cilindros alternativamente en sentido ascendente y descendente, atendiendo todas las peticiones por las que pasa[10]. Beneficia a los cilindros centrales, por lo que esta estrategia se combina con la de ubicar allí los bloques de acceso más frecuente.

***C-SCAN* o *C-LOOK* (algoritmos del ascensor de sentido único)**

Es una modificación del algoritmo del ascensor para evitar la discriminación de los cilindros. Ahora el cabezal recorre los cilindros en un único sentido (ascendente), volviendo al cilindro 0 (operación de **recalibrar**) entre cada recorrido.

5.3.5 Tratamiento de errores

Por su naturaleza mecánica, los discos están expuestos a numerosas causas de error. Es importante que las primitivas del núcleo del sistema operativo devuelvan un código que permita tratar el error para, si es posible, recuperar la operación.

Los errores más comunes en los accesos al disco se pueden englobar en alguno de los dos tipos siguientes:

- Errores de posicionamiento: el cabezal no se sitúa sobre el cilindro solicitado. Es necesario recalibrar, lo que supone poner el cabezal en el punto de referencia (cilindro 0), para poder reintentar el posicionamiento.
- Errores de control de paridad, que pueden indicar la existencia de un sector dañado o la presencia de suciedad en el cabezal o la superficie del disco.

En cualquier caso el gestor del disco tiene la opción de reintentar la operación un número razonable de veces. Si finalmente se consigue realizar la operación, el error se considera transitorio y el usuario no llega a tener noticias del percance (podría tratarse de una simple partícula de polvo que terminó desapareciendo). Si el

[10] Más precisamente, el algoritmo SCAN barre el disco hasta los cilindros extremos, mientras que el LOOK cambia el sentido tras atender la petición pendiente más alejada.

problema persiste (p. ej. si la causa fuera un sector dañado), el error se considera permanente y se transmite a los niveles superiores del sistema operativo. Los discos de hoy en día introducen mecanismos automáticos para tratar los errores permanentes, como la reubicación de sectores dañados. Una tabla hace corresponder cada sector lógico dañado a un sector físico en otra ubicación del disco. Obsérvese que esta técnica es análoga a la de interleaving, con la diferencia de que la correspondencia entre sectores lógicos y físicos es global a todo el disco.

5.3.6 Evaluación del rendimiento

Como se ha visto más arriba, el acceso y transferencia de un determinado bloque del disco conlleva una sucesión de operaciones con retardos asociados, algunos de ellos considerables al ser de naturaleza mecánica. Los tiempos que componen una operación de acceso a un bloque son los siguientes:

(1) Tiempo de posicionamiento sobre el cilindro, t_{pos}. Depende del recorrido que tenga que hacer el brazo de cabezales (distancia al cilindro desde la posición actual). Para la estimación del tiempo de acceso vamos a suponer un tiempo medio (que denotaremos T_{pos}), dependiente de la estrategia de planificación de accesos que se siga.

(2) Selección de pista. Se trata de seleccionar uno de los cabezales del brazo. Como únicamente supone la activación de una señal eléctrica, este retardo no se considera.

(3) Tiempo de posicionamiento en el comienzo del primer sector del bloque. Está relacionado con la velocidad de giro y varía desde 0 al tiempo de un giro completo, t_{giro}. En media hay que considerar este retardo como de $0{,}5t_{giro}$.

(4) Tiempo de posicionamiento en el resto de los sectores (para bloques de más de un sector). Los accesos a los demás sectores están condicionados por la posición del cabezal tras la transferencia del sector anterior, lo que depende del factor de interleaving o, alternativamente, de que el tamaño del buffer permita almacenar todos los sectores del bloque. En un disco de S sectores, sin buffers y con factor de interleaving F, el tiempo de posicionamiento en cada uno de los $N–1$ últimos sectores de un bloque de N sectores será Ft_{giro}/S.

(5) Tiempo de acceso a cada sector. Depende de la velocidad de giro y del número de sectores por pista. Para cada sector del bloque es t_{giro}/S.

(6) Tiempo de transferencia de sector, t_{trans}. Involucra una operación de DMA y depende de múltiples factores, como la velocidad del bus, la existencia o no de buffers en el controlador o el funcionamiento del DMA. Este tiempo se solapa

con el posicionamiento en el siguiente sector, de forma que sólo hay que considerar el tiempo de transferencia de un sector por bloque, independientemente de su tamaño.

Además, habría que considerar la incidencia de los retardos debidos al tratamiento de errores (por ejemplo, puede ser necesario posicionar más de una vez el cabezal), que depende de las probabilidades de cada tipo de error. Simplificando, para bloques de N sectores, con factor de interleaving F, sin considerar el tratamiento de errores, ni la existencia de buffers en el controlador, ni solapamiento entre operaciones, se obtiene la siguiente expresión del tiempo medio de acceso al bloque:

$$T_{acc} = T_{pos} + (0{,}5 + (N-1)F/S + N/S)\ t_{giro} + t_{trans}$$

Hay que insistir en que esta expresión responde a un modelo simplificado con las condiciones comentadas. En otros casos, la expresión del tiempo de acceso debe deducirse de acuerdo a los parámetros definidos para el sistema particular.

5.3.7 Esquema de un manejador de discos

Habitualmente, los programas de usuario acceden al disco indirectamente a través del sistema de ficheros. Las rutinas de entrada/salida resuelven la independencia del dispositivo, traducen los parámetros de la llamada al sistema a un número de bloque absoluto y, en su caso, ordenan al manejador del dispositivo el acceso al bloque. Los detalles de este proceso se verán en el capítulo de gestión de ficheros. El manejador del disco es el responsable de gestionar el acceso a los sectores correspondientes del disco. Resumiremos aquí el esquema de funcionamiento de un manejador de discos, particularizando el esquema general introducido en la Sección 5.2.2.3.

(1) Cuando el manejador detecta un evento de petición pendiente, toma un elemento (IORB) de su cola de peticiones y extrae los parámetros de la petición. Se traduce el número de bloque absoluto a los parámetros del disco: cilindro, pista y sectores. Si hubiera más de una petición pendiente, el manejador aplica una de las políticas de planificación de accesos estudiadas en la Sección 5.3.4.

(2) Si el motor estuviese parado, el manejador programa el encendido de la unidad y se bloquea por tiempo hasta que el disco ha alcanzado el régimen de revoluciones permanente[11].

[11] Como se vio anteriormente, esto es lo habitual en discos flexibles y ópticos. La parada de un disco rígido suele configurarse por motivos de ahorro de energía, tras un largo tiempo sin uso (del orden de la hora).

(3) Si el cabezal no estuviese colocado sobre el cilindro a acceder, se programa una operación de posicionar. El manejador queda bloqueado hasta la finalización de la operación. Como esta maniobra está sujeta a errores, es habitual, en su caso, reintentarla después de una operación de recalibrado.

(4) Para cada uno de los sectores del bloque, programa la transferencia DMA correspondiente, en uno u otro sentido dependiendo de la operación (lectura o escritura). Cada transferencia implica el bloqueo del manejador y la consiguiente comprobación de errores tras la finalización de la transferencia (señalada por la rutina de atención a la interrupción de DMA), con la posibilidad de reintento.

(5) El manejador escribe en el IORB el resultado de la operación.

(6) El manejador señala el evento especificado por la rutina de E/S en el IORB.

En el Ejercicio 6 se propone aplicar este esquema para programar un driver de disco.

5.4 Bibliografía

La gestión de la entrada/salida es un tema difícil de sistematizar, por lo que muchos textos se limitan a una exposición descriptiva de las características de los dispositivos, dedicando una mayor atención al disco como soporte del sistema de ficheros. Cabe destacar el esfuerzo de [LIS93], de donde hemos tomado el modelo de gestión de la Sección 5.2. Otros textos de interés pueden ser [STA05], por su tratamiento del almacenamiento intermedio y por los ejemplos de evaluación en el acceso a discos en sistemas comerciales, y [TAN08] y [TAN98] por la descripción de dispositivos en general. Este último describe además la implementación del sistema operativo MINIX.

5.5 Ejercicios

1 Considerar un sistema operativo donde la gestión de la entrada/salida sigue el modelo cliente-servidor, en comparación con otro donde el acceso a los recursos se gestiona con semáforos. Comparar ambos sistemas desde el punto de vista de la posibilidad de interbloqueos.

2 Cuando en un sistema como UNIX se cancela la ejecución de un programa (mediante ^C), es posible que se siga obteniendo la salida por pantalla del programa durante un tiempo. Explicar este efecto.

3 Dado un disco con una geometría de (C, P, S) = (300, 2, 40), bloques de 4 sectores y sectores de 512 bytes, ¿dónde se encuentra (cilindro, pista, sector) el bloque 155?

4 Un disco de doble cara posee 80 pistas por cara y 9 sectores de 512 bytes. El disco gira en la unidad a 600 r.p.m. El tiempo medio de posicionamiento del cabezal en una pista es de 20 ms. El controlador transfiere un sector mediante DMA por robo de ciclo en un tiempo medio de 5 ms. Tamaño de bloque: 1Kbyte.

(a) Calcular la cantidad de información que puede contener el disco, en Kbytes.

(b) Calcular el tiempo medio de acceso a un sector.

(c) Se quiere introducir interleaving software para transferir bloques de dos sectores en un tiempo mínimo. Calcular el factor de interleaving óptimo.

(d) Considerando el factor de interleaving óptimo, calcular el tiempo necesario para acceder a un bloque.

(e) Comparar este resultado con el obtenido si los bloques fueran de un sector. ¿Qué mejora de rendimiento se obtiene en cuanto a tiempo de acceso?

5 El driver de una unidad de discos de 80 cilindros acaba de atender una petición al cilindro 23. En la cola hay pendientes peticiones a los cilindros 17, 30, 2, 41, 77, 62, 41 y 53. Suponiendo que no llegan nuevas peticiones:

(a) Calcular las distancias totales que debe recorrer el cabezal utilizando respectivamente algoritmos de posicionamiento FIFO, SSF, SCAN y C-SCAN para atender las peticiones pendientes.

(b) Suponiendo un retardo medio de 1 ms por cilindro recorrido y que la operación de recalibrar tarda 40 ms, calcular los tiempos medios de posicionamiento para cada una de estas peticiones con los algoritmos SCAN y C-SCAN.

6 Considerar un sistema operativo por capas, de acuerdo al modelo de la Figura 5.1, donde la capa más interna, el núcleo, proporciona las funciones básicas de E/S y DMA, manejo de interrupciones, control y planificación de procesos, y sincronización. Sobre el núcleo se implementan los manejadores de dispositivos y las rutinas de E/S de acuerdo al modelo cliente-servidor. El núcleo proporciona, entre otras, las siguientes primitivas (sus nombres son autoexplicativos, y se ha añadido el sufijo *_nuc* para subrayar su carácter de interfaz del núcleo):

```
int quiensoy_nuc();
int wait_nuc(int semaforo);
int signal_nuc(int semaforo);
int encender_motor_nuc (int unidad);
int apagar_motor_nuc (int unidad);
int posicionar_cilindro_nuc (int unidad, int cilindro);
int recalibrar_nuc (int unidad);
int leer_sector_nuc (int unidad, int pista, int sector, char *buff);
int escribir_sector_nuc (int unidad, int pista, int sector, char *buff);
```

Sobre el núcleo se tienen las siguientes definiciones:

```
/* Descriptores de dispositivos */

struct dd {
  char nombre_driver[80];
  struct cola cola_driver;    /* cola donde encolar la petición */
  int sema_driver;            /* semáforo donde señalar la petición */
  ...
} drivers[MAX_DDS];

/* Definiciones para el driver de disco */

#define DRIVER_DISCO ...
int estado_disco;       /* PARADO, TRANSITORIO, EN_MARCHA */
int cilindro_actual;

/* IORBs */

struct iorb {
  struct item elemento_cola;
  int pid;
  int operacion;  /* LEER, ESCRIBIR, ... */
  int IO_dir;
  char *pbuff;
  int longitud;
  int evento;                 /* semáforo donde esperar respuesta */
  int diagnostico;            /* a rellenar por el driver */
} peticiones[MAX_IORBS];
```

Disponemos de una rutina de E/S cliente síncrona para el acceso al driver del disco:

```
int peticion_disco (int op, int bloque, char *pbuff, int ev)
{
  int id, r;
  struct iorb *pet;

  id= quiensoy_nuc();
  pet= reservar_iorb();
  pet->pid= id;
  pet->operacion= op;
  pet->IO_dir= bloque;
  pet->pbuff= pbuff;
  pet->evento= ev;
  encolar_iorb(&drivers[DRIVER_DISCO].cola_driver, pet);
  signal_nuc(drivers[DRIVER_DISCO].sema_driver);
  wait_nuc(ev);
  r= pet->diagnostico;
  liberar_iorb(pet);
  return(r);
}
```

Con estas definiciones, se pide:

(a) Construir un driver de disco básico que trate bloques de un sector y no realice ningún tipo de control de errores (simplemente devuelve el error a la rutina de E/S cliente cuando este se produzca). La geometría del disco es C·P·S.

(b) Modificar el driver anterior para introducir control de errores. Ahora, un error de posicionado se tratará recalibrando el cabezal y reintentando el posicionamiento. Los errores de acceso se tratarán mediante reintento. Si tras tres intentos el error persiste, se notificará a la rutina de E/S.

(c) Modificar el driver anterior para tratar bloques de 2 sectores. Se supone interleaving hardware.

(d) ¿Qué modificaciones habría que hacer sobre este driver para gestionar las peticiones mediante una política tipo C-SCAN?

(e) ¿Qué modificaciones habría que hacer en la rutina de E/S para que esta fuera asíncrona?