ESCUELA DE PREPARACIÓN DE OPOSITORES
E. P. O.
C/. La Merced, 8 – Bajo A Telf.: 968 24 85 54
30001 MURCIA

INF18 – SAI18

Sistemas operativos: gestión de entradas/salidas.

Esquema.

1 Introducción.

Una de las funciones principales de un sistema operativo es el control de todos los dispositivos de entrada/salida de la computadora. Debe enviar comandos a los dispositivos, detectar las interrupciones y controlar los errores. También debe proporcionar una interfaz entre los dispositivos y el resto del sistema, sencilla y fácil de usar. En la medida de lo posible, la interfaz debe ser la misma para todos los dispositivos (independencia del dispositivo).

La forma de administrar la E/S por parte del sistema operativo es el tema de nuestro estudio. En primer lugar, analizaremos algunos de los principios del hardware de E/S y después del software de E/S en general. Después, estudiaremos con detalle tres dispositivos comunes de E/S: discos, relojes y terminales, para concluir nuestro estudio con una visión global de la E/S en los sistemas operativos UNIX, MS-DOS y Windows 2000.

2 Principios del hardware de E/S.

2.1 Dispositivos de E/S.

Los dispositivos de E/S se pueden dividir en dos categorías: dispositivos de bloque y dispositivos de carácter. Un dispositivo de bloque almacena la información en bloques de tamaño fijo, cada uno con su propia dirección. La propiedad esencial de un dispositivo de bloque es la posibilidad de leer o escribir en un bloque de forma independiente de los demás. En otras palabras, en todo momento, el programa puede leer o escribir en cualquiera de los bloques. Los discos son dispositivos de bloque.

Un dispositivo de carácter envía o recibe un flujo de caracteres, sin sujetarse a una estructura de bloques. No se pueden utilizar direcciones ni tienen una operación de búsqueda. Los terminales, impresoras de línea, interfaces de una red y muchos otros dispositivos no parecidos a los discos son dispositivos de carácter.

Este esquema de clasificación no es perfecto. Por ejemplo, los relojes no tienen direcciones por medio de bloques. Tampoco generan o aceptan flujos de caracteres. Lo único que hacen es provocar interrupciones a intervalos bien definidos. Aun así, el modelo de dispositivos de bloque y de carácter es lo bastante general para ser utilizado como base para trabajar el software de sistemas operativos de forma independiente del dispositivo de E/S.

2.2 Controladores de dispositivos.

Las unidades de E/S constan por lo general de un componente mecánico y otro electrónico. El componente electrónico se llama controlador del dispositivo o adaptador. El componente mecánico es el propio dispositivo. Mencionamos esa distinción entre controlador y dispositivo porque el sistema operativo casi siempre trabaja con el controlador y no con el dispositivo. La interfaz entre el controlador y el dispositivo es con frecuencia de muy bajo nivel.

La labor del controlador es convertir el flujo de bits serie en un bloque de bytes y llevar a cabo cualquier corrección de errores necesaria. Lo común es que el bloque de bytes se ensamble, bit a bit, en un buffer dentro del controlador. Después de verificar la suma y declarar al bloque libre de errores, se le puede copiar en la memoria principal.

Cada controlador tiene unos cuantos registros que utiliza para la comunicación con la CPU. En ciertas computadoras, estos registros son parte del espacio normal de

direcciones de la memoria. Este esquema se llama E/S mapeada a memoria. Otras computadoras utilizan un espacio de direcciones especial para la E/S, asignando a cada controlador una parte de él.

El sistema operativo realiza la E/S al escribir comandos en los registros de los controladores. Al aceptar un comando, la CPU puede dejar al controlador y dedicarse a otro trabajo. Al terminar el comando, el controlador provoca una interrupción para permitir que el sistema operativo obtenga el control de la CPU y verifique los resultados de la operación. La CPU obtiene los resultados y el estado del dispositivo al leer uno o más bytes de información de los registros del controlador.

2.3 Acceso directo a memoria (DMA).

Muchos controladores permiten el acceso directo a memoria o DMA. Para explicar el funcionamiento del DMA, analicemos primero la forma en que se lee el disco si no se utiliza éste. En primer lugar, el controlador lee en serie el bloque de la unidad, bit a bit, hasta que todo el bloque se encuentra en el buffer interno del controlador. A continuación, calcula la suma de verificación para corroborar que no existen errores de lectura. Entonces, el controlador provoca una interrupción. Cuando el sistema operativo empieza su ejecución, puede leer el bloque del disco por medio del buffer del controlador, un byte o una palabra a la vez, en un ciclo, en el que durante cada iteración se lee un byte o una palabra del registro del controlador y se almacena en memoria.

DMA se ideó para liberar a la CPU de este trabajo de bajo nivel. Al utilizarlo, la CPU le proporciona al controlador dos elementos de la información, además de la dirección del bloque en el disco: la dirección en memoria donde debe ir el bloque y el número de bytes a transferir.

Después de que el controlador ha leído todo el bloque del dispositivo a su buffer y ha corroborado la suma de verificación, copia el primer byte o palabra a la memoria principal, en la dirección especificada por medio de la dirección de memoria de DMA. Entonces incrementa la dirección DMA y decrementa el contador DMA en el número de bytes que acaba de transferir. Este proceso se repite hasta que el contador se anula, momento en el cual el controlador provoca una interrupción. Al iniciar su ejecución el sistema operativo, no tiene que copiar el bloque en la memoria: ya se encuentra ahí.

Aunque los datos se transfieren del controlador a la memoria, ya sea mediante la CPU o mediante el propio controlador, el siguiente sector pasará debajo de la cabeza del disco y los bits llegarán al controlador. Los controladores simples no pueden enfrentarse a la E/S simultánea, de forma que mientras se lleva a cabo una transferencia en la memoria, el sector que pasa debajo de la cabeza del disco se pierde. Como resultado, el controlador sólo podrá leer hasta el siguiente bloque. La lectura de una pista completa se realizará entonces en dos rotaciones completas, una para los bloques impares y otra para los pares. Si el tiempo necesario para una transferencia de un bloque del controlador a la memoria por medio del bus es mayor que el tiempo necesario para leer un bloque del disco, podría ser necesario leer un bloque y después saltar dos (o más) bloques.

El salto de bloques, que se ejecuta para darle tiempo al controlador para la transferencia de los datos a la memoria se llama separación. Al dar formato al disco, los bloques se numeran tomando en cuenta el factor de separación. La idea de numerar los bloques de esta manera es permitir al sistema operativo que lea los bloques con numeración consecutiva y conserve la máxima velocidad posible del hardware.

3 Principios del software de E/S.

La idea básica es organizar el software como una serie de capas, donde las capas inferiores oculten las peculiaridades del hardware a las capas superiores de forma que éstas se preocupen por presentar una interfaz agradable, limpia y regular a los usuarios.

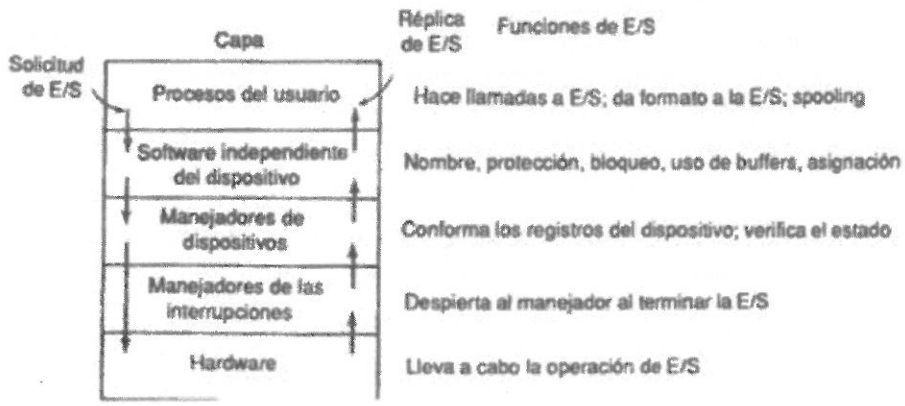
Un concepto clave en el diseño del software de E/S es la independencia del dispositivo. El sistema operativo debe encargarse de los problemas causados por el hecho de que los dispositivos sean distintos y requieran directivas diferentes. El objetivo de lograr nombres uniformes está relacionado con el de independencia del dispositivo. El nombre de un archivo o dispositivo debe ser sólo una cadena o un entero y no debe depender del dispositivo. En UNIX, todos los discos se pueden integrar juntos en la jerarquía del sistema de archivos de maneras arbitrarias, de forma que el usuario no tenga que tener conciencia del nombre de cada dispositivo.

Otro aspecto importante del software de E/S es el manejo de errores. En general, los errores deben manejarse lo más cerca posible del hardware. Si el controlador descubre un error de lectura, debe tratar de corregirlo. Si no puede corregirlo, debería controlarlo, tal vez mediante la forma de leer el bloque de nuevo. Muchos errores son momentáneos, tales como los de lectura provocados por partículas de polvo en la cabeza de lectura y desaparecen al repetirse la operación. Sólo en el caso en que los niveles inferiores no puedan resolver el problema, se informa a los niveles superiores.

Otro de los aspectos claves son las transferencias síncrona (por bloques) o asíncrona (controlada por interruptores). La mayor parte de la E/S es asíncrona, pero los programas son más fáciles de escribir si las operaciones de E/S son por medio de bloques (después de un comando READ, el programa se suspende hasta que los datos estén disponibles en el buffer). El sistema operativo se encarga de hacer que las operaciones controladas por interruptores parezcan del tipo de bloques para el usuario.

El concepto final que analizaremos es la comparación de los dispositivos que se pueden compartir y los dispositivos de uso exclusivo. Algunos de los dispositivos de E/S, como los discos, pueden ser utilizados por varios usuarios al mismo tiempo. Otros dispositivos, como las impresoras, deben dedicarse a un solo usuario hasta concluir con él. El uso de dispositivos de uso exclusivo presenta también una serie de dificultades. De nuevo, el sistema operativo debe administrar los dispositivos compartidos y de uso exclusivo de forma que evite dichos problemas.

Estos objetivos se logran al estructurar el software de E/S en cuatro capas: manejadores de interrupciones, directivas de dispositivos, software de sistema operativo independiente de los dispositivos, y software a nivel usuario.



3.1 Manejadores de interrupciones.

Las interrupciones deben ocultarse en lo más profundo del sistema operativo, de modo que sólo una pequeña parte del sistema sepa de ellas. La mejor forma de esconderlas es que cada proceso que inicie una operación de E/S se bloquee hasta que termine la E/S y ocurra la interrupción. Al ocurrir la interrupción, el procedimiento de interrupción realiza lo debido para eliminar el bloqueo del proceso que lo inició.

3.2 Manejadores de dispositivos.

La labor de un manejador de dispositivo es aceptar las solicitudes abstractas del software independiente del dispositivo y verificar la ejecución de dicha solicitud. Una solicitud común es la de leer el bloque *n*. Si el manejador está inactivo al recibir la solicitud, lleva a cabo la solicitud de manera inmediata. Sin embargo, si se encuentra ocupado con una solicitud, normalmente general formará la nueva solicitud en una cola de solicitudes pendientes, para darles paso tan pronto sea posible.

El primer paso para llevar a cabo una solicitud de E/S para un disco, por ejemplo, es traducirla de términos abstractos a términos concretos. Para un manejador de disco, esto quiere decir calcular el lugar donde se encuentra el bloque solicitado, verificar si el motor de la unidad funciona, si el brazo está colocado en el cilindro adecuado, etc. En resumen, debe decidir cuáles son las operaciones necesarias del controlador y su orden.

Una vez determinados los comandos a enviar al controlador, los envía al escribir en los registros de dispositivo del controlador. Algunos controladores pueden manejar un solo comando a la vez. Otros pueden aceptar una lista de comandos, los cuales pueden llevar a cabo por sí mismos, sin ayuda posterior del sistema operativo.

Una vez concluida la operación de E/S, el manejador debe verificar los errores. Si todo está en orden, el manejador dispondrá de datos para transferirlos al software independiente del dispositivo. Por último, devuelve información de estado para informar de los errores a quien lo llamó. Si existen otras solicitudes pendientes, debe seleccionar e iniciar alguna de ellas. Si no hay pendientes, el manejador se bloquea en espera de una solicitud.

3.3 Software de E/S independiente del dispositivo.

La función básica del software independiente del dispositivo es llevar a cabo las funciones de E/S comunes a todos los dispositivos, además de proporcionar una interfaz uniforme del software a nivel usuario.

Un aspecto fundamental en un sistema operativo lo forman los nombres de los archivos y los dispositivos de E/S. El software independiente del dispositivo asocia los nombres simbólicos de los dispositivos con el nombre adecuado. En UNIX, un nombre de dispositivo determina de manera única el nodo-i de un archivo especial, y este nodo-i contiene el número principal del dispositivo que se utiliza para localizar el manejador apropiado. El nodo-i contiene también el número secundario del dispositivo, que se transfiere como parámetro al manejador para determinar la unidad por leer o escribir.

Un aspecto muy relacionado con los nombres es el de la protección. En ciertos sistemas, como MS-DOS, no existe protección alguna. UNIX utiliza un esquema más flexible. Los archivos especiales correspondientes a los dispositivos de E/S están protegidos por los bits *rwx*. El administrador del sistema puede establecer entonces los permisos adecuados para cada dispositivo.

Los distintos discos pueden tener diferentes tamaños de sector. Es labor del software independiente del dispositivo ocultar este hecho y proporcionar un tamaño uniforme de los bloques a los niveles superiores; por ejemplo, al considerar varios sectores como un solo bloque lógico. De esta forma, las capas superiores sólo trabajan con dispositivos abstractos que utilizan el mismo tamaño de bloque lógico, de forma independiente al tamaño de sector físico.

El almacenamiento en buffers también es otro aspecto a considerar. Para los dispositivos de bloque, el hardware insiste por lo general en la lectura y escritura de bloques a la vez, pero los procesos del usuario son libres de leer y escribir en unidades arbitrarias. Si un proceso del usuario escribe la mitad de un bloque, el sistema operativo mantendrá los datos hasta que se escriban el resto de ellos, en cuyo momento el bloque podrá pasar al disco. Para los dispositivos de carácter, los usuarios pueden escribir los datos al sistema de manera más rápida que la velocidad con la que pueden salir, por lo que se requiere el uso de buffers. De igual modo la entrada del teclado puede llegar antes de que sea necesaria, lo cual también requiere buffers.

Al crear un archivo y llenarlo de datos, los nuevos bloques del disco deben ser asignados al archivo. Para esto, el sistema operativo necesita una lista o un mapa de bits de los bloques libres del disco, pero el algoritmo para localizar un bloque libre es independiente del dispositivo y se puede realizar por encima de la capa del manejador.

Algunos dispositivos, como las unidades de cinta magnética, sólo se pueden utilizar por un proceso en un determinado momento. El sistema operativo se encarga de examinar las solicitudes para el uso del dispositivo y aceptarlas o rechazarlas según la disponibilidad del dispositivo solicitado.

El manejo de errores lo realizan los manejadores. La mayoría de los errores dependen del dispositivo, por lo que sólo éste sabe qué hacer. Un error común se produce cuando un bloque del disco ha sido dañado y no se puede leer más. Después de que el manejador ha intentado leer el bloque un cierto número de veces, se da por vencido e informa del hecho al software independiente del dispositivo. A partir de ese momento, la forma de manejo del error es independiente del dispositivo. Si el error ocurrió durante la lectura de un archivo del usuario, podría ser suficiente informar del error a quien hizo la llamada. Sin embargo, si ocurrió durante la lectura de una estructura de datos crítica para el sistema, el sistema operativo podría no tener más opción que imprimir un mensaje de error y terminar.

3.4 Software de E/S en el espacio del usuario.

Aunque la mayoría del software de E/S está dentro del sistema operativo, una pequeña parte de él consta de librerías enlazadas con los programas del usuario e incluso programas completos que se ejecutan dentro del núcleo. Las llamadas al sistema de E/S, en general, son llevadas a cabo por los procedimientos de librería. La colección de todos estos procedimientos de librería es parte del sistema de E/S.

Algunos de estos procedimientos hacen un poco más que poner sus parámetros en el lugar apropiado para la llamada al sistema, mientras que otros realmente efectúan cierto trabajo. En particular, el formato de la entrada y la salida son realizados por medio de procedimientos de librería.

No todo el software de E/S a nivel usuario consta de procedimientos de librería. Otra categoría importante es el sistema de spooling. El spooling es una forma de trabajar con los dispositivos de E/S de uso exclusivo en un sistema de

multiprogramación. Consideremos un dispositivo que por lo general utiliza el spooling: la impresora de líneas. Aunque es fácil dejar que los procesos del usuario abran el archivo especial de carácter correspondiente a la impresora, supongamos que un proceso lo abrió y después no llevó a cabo actividad alguna durante horas. Ningún proceso podría imprimir nada.

En vez de esto, lo que hace es crear un proceso especial, llamado demonio y un directorio especial, llamado directorio de spooling. Para imprimir un archivo, un proceso genera en primer lugar todo el archivo a imprimir y lo pone en el directorio de spooling. El demonio, que es el único proceso con permiso para utilizar el archivo especial de la impresora, debe imprimir los archivos en el directorio. Al proteger el archivo especial contra el uso directo de los usuarios, se elimina el problema de tener a alguien manteniendo algún archivo abierto un largo tiempo innecesario.

4 Discos.

Todos los discos están organizados mediante cilindros, cada uno de los cuales contiene tantas pistas como cabezas apiladas verticalmente. Las pistas se dividen en sectores y el número usual de sectores en torno de la circunferencia es de ocho a 32. Todos los sectores contienen el mismo número de bytes, aunque los sectores cercanos a la orilla del disco serán mayores físicamente que los cercanos al anillo. El espacio adicional no se utiliza.

Una característica que tiene importantes implicaciones para el manejador del disco es la posibilidad de que un controlador realice búsquedas en una o más unidades al mismo tiempo. Mientras el controlador y el software esperan a que termine una búsqueda en una unidad, el controlador puede iniciar una búsqueda en otra. Muchos controladores pueden también leer o escribir en una unidad mientras buscan en otras, pero ninguno puede leer o escribir en dos unidades al mismo tiempo.

4.1 Algoritmos de programación del brazo del disco.

El tiempo para la lectura o escritura de un bloque de disco está determinado por tres factores: el tiempo de búsqueda (tiempo necesario para mover el brazo hasta el cilindro apropiado), el retraso rotacional (tiempo necesario para que el sector adecuado rote debajo de la cabeza) y el tiempo real de la transferencia. Para la mayoría de los discos, el tiempo de búsqueda domina, por lo que la reducción del tiempo promedio de búsqueda puede mejorar en gran medida el rendimiento del sistema.

Si el manejador del disco acepta solicitudes, una a la vez, y las lleva a cabo en ese orden (es decir, el primero en llegar es el primero en ser atendido o FCFS), poco se puede hacer para reducir el tiempo de búsqueda. Sin embargo, es posible otra estrategia si el disco está demasiado cargado. Es probable que cuando el brazo realiza una búsqueda a cargo de una solicitud, otros procesos pueden generar otras solicitudes al disco. Muchos manejadores de discos tienen una tabla, cuyo índice es el número de cilindro, con todas las solicitudes pendientes para cada cilindro enlazadas entre sí en una lista enlazada encabezada por los datos de la tabla.

Dado este tipo de estructura de datos, podemos mejorar el algoritmo de planificación "primero en llegar, primero en ser atendido". Para esto, consideremos un disco con 40 cilindros. Se presenta una solicitud para leer un bloque en el cilindro 11. Mientras se busca el cilindro, llegan nuevas solicitudes para los cilindros 1, 36, 16, 34, 9 y 12, en ese orden. Estos valores entran en la tabla de solicitudes pendientes, con una lista enlazada aparte para cada cilindro. Cuando termina la solicitud activa (para el

cilindro 11), el manejador del disco tiene la opción de elegir la siguiente solicitud a la cual dar paso. Por medio del algoritmo anterior, podría pasar al cilindro 1, después al 36, etc. Este algoritmo necesitaría movimientos del brazo de 10, 35, 20, 18, 25 y 3 respectivamente, para un total de 111 cilindros.

Otra alternativa es manejar siempre a continuación la solicitud más cercana, con el fin de minimizar el tiempo de búsqueda. Con las solicitudes anteriores, la secuencia es 12, 9, 16, 1, 34 y 36. Con esta secuencia, los movimientos del brazo son 1, 31 71 15, 33 y 2, para un total de 61 cilindros. Este algoritmo, primero la búsqueda más corta o SSF (*shortest seek first*) reduce a la mitad el número de movimientos del brazo en comparación con FCFS.

Por desgracia, SSF tiene un problema. Con un disco muy cargado, el brazo tenderá a permanecer a la mitad del disco la mayoría del tiempo, de modo que las solicitudes de los extremos deberán esperar hasta que una fluctuación estadística en la carga provoque que no existan solicitudes cerca de la mitad del disco. Las solicitudes lejanas de la mitad obtendrán un mal servicio.

La planificación de un ascensor en un edificio es similar al de la planificación del brazo de un disco. Las solicitudes llegan de manera continua y llaman al ascensor desde los diversos pisos (cilindros) aleatoriamente. El microprocesador que controla el ascensor puede mantener un registro de la secuencia en que los clientes oprimen el botón de llamada y darles servicio mediante FCFS, o bien utilizar SSF.

La mayoría de los ascensores se mantienen en movimiento en la misma dirección, hasta que no tienen más solicitudes pendientes en esa dirección; es entonces cuando cambian de dirección. Este algoritmo, conocido como el algoritmo del ascensor, necesita que el software conserve 1 bit: el de la dirección actual, ARRIBA y ABAJO. Al terminar con una solicitud, el manejador del disco o ascensor verifica el bit. Si indica ARRIBA, el brazo o cabina se desplaza hacia la siguiente solicitud pendiente hacia arriba, si ésta existe. Si no existen solicitudes pendientes en posiciones más altas, se invierte el bit de dirección. Cuando el bit indica ABAJO, el movimiento es hacia la siguiente posición inferior solicitada, si ésta existe.

Si aplicamos el algoritmo del ascensor con las siete solicitudes anteriores y el bit de dirección tiene un valor inicial de ARRIBA, el orden de servicio a los cilindros es 12, 16, 34, 36, 9 y 1, lo cual significa movimientos del brazo de 1, 4, 18, 2, 27 y 8, para un total de 60 cilindros. En este caso, el algoritmo del ascensor es un poco mejor que SSF, aunque generalmente es peor. Una propiedad interesante del algoritmo del ascensor es que dada cualquier colección de solicitudes, la cuota máxima del total de movimientos está fija: es el doble del número de cilindros.

Una ligera modificación de este algoritmo es rastrear siempre en la misma dirección. Después de darle servicio al cilindro con el número mayor, el brazo pasa al cilindro de número menor con una solicitud pendiente y continúa su movimiento hacia arriba.

Algunos controladores de disco proporcionan una forma para que el software inspeccione el número del sector activo debajo de la cabeza. Con uno de estos controladores, es posible otro tipo de optimización. Si dos o más solicitudes para el mismo cilindro están pendientes, el manejador puede enviar una solicitud para el sector que pasará por debajo de la cabeza. Cuando existen varias pistas en un cilindro, se pueden hacer solicitudes consecutivas de distintas pistas, sin que esto genere un problema. El controlador puede seleccionar cualquiera de sus cabezas de manera

instantánea, puesto que la selección de la cabeza no implica movimientos del brazo o retrasos debido a la rotación.

Cuando existen varias unidades, se debe tener una tabla de solicitudes pendientes para cada unidad. Siempre que una unidad esté inactiva, deberá hacerse una búsqueda para mover su brazo hacia el cilindro siguiente necesario. Cuando termina la transferencia actual, se puede hacer una verificación para ver si las unidades están en la posición del cilindro correcto. Si una o más unidades están en este caso, se puede iniciar la siguiente transferencia en una unidad que ya se encuentre en el cilindro correcto. Si ninguno de los brazos está en el lugar correcto, el manejador debe realizar una nueva búsqueda en la unidad que terminó la transferencia y esperar hasta la siguiente interrupción para ver cuál de los brazos llega primero a su destino.

Al mejorar la tecnología de los discos, los tiempos de búsqueda se acortan, pero no se modifica el retraso debido a la rotación. En algunos discos, el tiempo promedio de búsqueda ya es más corto que el retraso rotacional. Si continúa esta tendencia, el algoritmo del ascensor será obsoleto, puesto que el retraso rotacional será el factor dominante.

4.2 Manejo de errores.

Los discos están sujetos a una amplia gama de errores. Algunos de los más comunes son: error de programación (por ejemplo, solicitar un sector no existente), error en la suma de verificación (por ejemplo, provocado por polvo en la cabeza o por un bloque del disco dañado en forma física), error de búsqueda (por ejemplo, el brazo se envía al cilindro 6 pero va al 7), y error del controlador (por ejemplo, él controlador no acepta los comandos). El manejador del disco debe controlar cada uno de estos errores de la mejor manera posible.

Los errores de programación ocurren cuando el manejador le dice al controlador que busque un cilindro no existente, lea un sector no existente, utilice una cabeza no existente o transfiera desde o hacia una memoria no existente. Lo único que puede hacer el manejador es terminar con la solicitud del disco actual mediante un error y esperar que éste no aparezca con demasiada frecuencia.

Los errores temporales en la suma de verificación son provocados por partículas de polvo en el aire, que penetran entre la cabeza y la superficie del disco. La mayor parte del tiempo podrán eliminarse al repetir la operación unas cuantas veces. Si persiste el error, el bloque puede ser marcado como un bloque defectuoso de forma que el software lo evite. Otra alternativa es reservar unas cuantas pistas que por lo general no estén disponibles para los programas del usuario. Al dar formato a una unidad de disco, el controlador determina los bloques defectuosos y sustituye automáticamente una de las pistas de repuesto por la pista defectuosa. La tabla que asocia las pistas defectuosas con las pistas de repuesto se mantiene dentro de la memoria interna del controlador y en el disco. Esta sustitución es transparente (invisible) para el manejador, excepto que el algoritmo del ascensor podría obtener un mal rendimiento si el controlador utiliza en forma secreta el cilindro 800 cada vez que se solicite el cilindro 3.

Los errores de búsqueda son provocados por problemas mecánicos en el brazo. El controlador lleva un registro interno de la posición del brazo. Para realizar una búsqueda, envía una serie de pulsaciones al motor del brazo, un pulso por cada cilindro, para desplazar el brazo hacia el nuevo cilindro. Cuando el brazo llega a su destino, el controlador lee el número del cilindro activo (escrito al dar formato a la unidad) si el brazo está en un lugar equivocado, ocurre un error de búsqueda. Algunos controladores

corrigen de manera automática los errores de búsqueda, pero otros sólo activan un bit de error y dejan el resto al manejador.

4.3 Discos RAM.

Un disco RAM utiliza una parte de la memoria principal asignada con anterioridad para almacenar los bloques. Un disco RAM tiene la ventaja del acceso instantáneo, lo que lo hace adecuado para el almacenamiento de programas o datos con acceso frecuente.

El disco RAM se divide en *n* bloques. Cada bloque tiene el mismo tamaño que el de bloque de los discos reales. Cuando el manejador recibe un mensaje para la lectura o escritura en un bloque, sólo calcula el lugar de la memoria del disco en RAM donde se encuentra el bloque solicitado y lee o escribe en él, en vez de utilizar un disco flexible o un disco duro. Por lo general, se puede hacer la transferencia al llamar un procedimiento en lenguaje ensamblador, el cual copia desde o hacia el programa del usuario a la máxima velocidad posible para el hardware.

5 Relojes.

Los relojes son esenciales para la operación de cualquier sistema con tiempo compartido. El software para reloj toma la forma de un manejador de dispositivo, aunque un reloj no es un dispositivo de bloque, ni de carácter.

Los relojes más sencillos están sujetos a la línea de corriente de 110 ó 220 voltios y provocan una interrupción por cada ciclo de voltaje. Otro tipo de relojes constan de tres componentes: un oscilador de cristal, un contador y un registro. Cuando se corta una pieza de cristal de cuarzo y se monta en una estructura bajo tensión, se puede lograr que genere una señal periódica de muy alta precisión. Esta señal se alimenta en el contador para que cuente descendente hasta cero. Cuando el contador llega a cero, provoca una interrupción de la CPU.

Los relojes programables tienen varios modos de operación. En el modo de una instancia, cuando el reloj se inicializa, copia el valor del registro en el contador y después decrementa el contador en cada pulso del cristal. Cuando el contador llega a cero, provoca una interrupción y se detiene hasta que es inicializado de nuevo de forma explícita por el software. En el modo de onda cuadrada, después de llegar a cero y provocar la interrupción, el registro se copia automáticamente en el contador y todo el programa se repite de forma indefinida. Estas interrupciones periódicas se llaman marcas del reloj.

La ventaja del reloj programable es que su frecuencia de interrupción puede ser controlada por el software. Si se utiliza un cristal de 1 MHz, entonces el contador se pulsará cada microsegundo. Con registros de 16 bits, las interrupciones se pueden programar para que ocurran a razón de 1 a 65535 microsegundos. Los chips de reloj programable contienen por lo general dos o tres relojes programables de forma independiente.

5.1 Software para relojes.

Lo único que hace el hardware para relojes es generar las interrupciones a intervalos dados. Todo lo restante relativo al tiempo es realizado por el software, el manejador del reloj. Las labores precisas del manejador del reloj varían entre los distintos sistemas, pero por lo general están incluidas las siguientes: mantener la hora

del día, evitar que los procesos se ejecuten durante más tiempo del permitido, mantener un registro del uso de la CPU, controlar la llamada al sistema ALARM por parte de los procesos del usuario, proporcionar cronómetros guardianes de partes del propio sistema, y realizar resúmenes, monitorizar y recolectar estadísticas.

Conservar la hora del día no es difícil. Sólo requiere incrementar un contador en cada marca del reloj. Lo único que hay que vigilar es el número de bits en el contador de la hora del día. A razón de 60 Hz, un contador de 32 bits tendrá un desbordamiento después de casi dos años. Se utilizan tres puntos de vista para resolver este problema. La primera forma es utilizar un contador de 64 bits, aunque esto hace que la adición de una unidad al contador sea una operación más cara, puesto que deberá realizarse más veces por cada segundo. En la segunda, se mantiene la hora del día en segundos, mediante un contador secundario que cuenta las marcas hasta acumular un segundo. Puesto que 2^{32} segundos son más de 136 años, este método funcionará bastante tiempo. La tercera vía es contar por medio de marcas, pero de forma relativa al momento en que arrancó el sistema, en vez de referirse a cierto hecho externo. Cuando el usuario escribe el tiempo real, el tiempo de arranque del sistema se calcula a partir del valor actual de la hora del día y se almacena en la memoria en forma conveniente. Más adelante, al solicitar la hora del día, la hora almacenada se añade al contador para obtener la hora actual.

La segunda función del reloj es evitar que los procesos se ejecuten durante mucho tiempo. Al iniciar un proceso, el planificador debe inicializar un contador con el valor del quantum de ese proceso en marcas de reloj. En cada interrupción del reloj, el manejador del reloj decrementa el contador del quantum en 1. Cuando toma el valor 0, el manejador del reloj llama al planificador para iniciar otro proceso.

La tercera función del reloj es contabilizar el uso de la CPU. La forma más precisa de hacer esto es comenzar con un segundo cronómetro, distinto del cronómetro principal del sistema, siempre que se inicie un proceso. Cuando ese proceso se detiene, se puede leer un cronómetro para determinar el tiempo durante el cual se ha ejecutado el proceso. El segundo cronómetro debe ser resguardado al ocurrir una interrupción y ser restaurado después de ésta. Una forma menos exacta, pero más sencilla es mantener como variable global un puntero a la entrada de la tabla de procesos para el proceso en ejecución. En cada marca de reloj, se incrementa un campo en la entrada del proceso activo. De esta forma, cada marca del reloj se "carga" al proceso en ejecución al momento de la marca. Un problema menor con esta estrategia es que si ocurren varias interrupciones durante la ejecución de un proceso, se le cargará una marca completa, aunque no haya realizado mucho trabajo. La contabilidad precisa del uso de la CPU durante las interrupciones es demasiado cara y nunca se lleva a cabo.

En UNIX y otros sistemas, un proceso puede solicitar al sistema operativo que dé una advertencia después de cierto intervalo de tiempo. La advertencia es por lo general una señal, una interrupción, un mensaje o algo similar. Si el manejador del reloj tiene relojes suficientes, podría utilizar un reloj independiente para cada solicitud. Si éste no es el caso, debe simular varios relojes virtuales con un único reloj físico. Una forma de lograr esto es tener una tabla con el tiempo de señalización para todos los cronómetros pendientes, así como una variable que indique el tiempo de la siguiente. Al actualizar la hora del día, el manejador verifica si ha ocurrido la señal más cercana. En ese caso, se busca en la tabla la siguiente señal por ocurrir. Si se esperan muchas señales, es más eficiente simular varios relojes mediante una cadena de todas las solicitudes pendientes, ordenadas según el tiempo, en una lista enlazada.

Durante una interrupción del reloj, el manejador del mismo debe: incrementar el tiempo real, decrementar el quantum y verificar si es 0, hacer la contabilidad de la CPU y decrementar el contador de la alarma. Cada una de estas operaciones debe ser muy rápida, puesto que debe repetirse varias veces en cada segundo.

Las partes del sistema también necesitan establecer sus cronómetros. Estos se llaman cronómetros guardianes. Por ejemplo, para utilizar un disco flexible, el sistema debe activar el motor y esperar cerca de 500 mseg hasta lograr la velocidad adecuada. Al terminar la E/S, es una buena idea iniciar un cronómetro guardián y desactivar el motor sólo en el caso que no se realice operación alguna de E/S durante, por ejemplo, 3 segundos, para evitar el retraso de 500 mseg en cada operación. El mecanismo utilizado por el manejador del reloj para el control de los cronómetros guardianes es igual al de las señales de usuario. La única diferencia es que al agotar su tiempo un cronómetro, no provoca una señal, sino que el manejador llama un procedimiento proporcionado por quien hizo la llamada.

Algunos sistemas operativos proporcionan un mecanismo por medio del cual un programa del usuario puede lograr que el sistema construya un histograma del contador del programa, para poder ver dónde se invierte el tiempo. Si se pueden hacer los resúmenes, el manejador verifica en cada marca si el proceso es monitorizado y, en caso afirmativo, calcula el número binario correspondiente al contador del programa activo. Entonces incrementa ese número en 1. Este mecanismo también se puede utilizar para monitorizar el propio sistema.

6 Terminales.

Cada computadora tiene uno o más terminales que se utilizan para comunicarse con él. Los terminales tienen un gran número de formas distintas. El manejador del terminal se encarga de ocultar todas estas diferencias, de forma que la parte independiente del dispositivo en el sistema operativo y los programas del usuario no tienen que volverse a escribir para cada tipo de terminal.

6.1 Hardware para terminales.

Los terminales se pueden clasificar en dos grandes categorías en base a la forma en que el sistema operativo se comunica con ellos. La primera categoría consta de las terminales con interfaz por medio del estándar RS-232; la segunda categoría consta de terminales mapeados a memoria.

Los terminales RS-232 son dispositivos con un teclado y un monitor que se comunican mediante una interfaz serie. Para enviar un carácter a un terminal RS-232, el ordenador debe transmitir 1 bit a la vez, comenzando con un bit de inicio y seguido por 1 ó 2 bits finales para delimitar el carácter.

Puesto que tanto las computadoras como los terminales trabajan internamente con caracteres completos, pero deben comunicarse en una línea serie, se han desarrollado chips para realizar las conversiones carácter/serie y viceversa. Se les llama UART (siglas en inglés de Transmisores-Receptores Asíncronos Universales).

Para imprimir un carácter, el manejador del terminal escribe éste en la tarjeta de interfaz, donde se guarda en un buffer y después la UART lo desplaza a lo largo de la línea serie, un bit a la vez. Como resultado de esta lenta velocidad de transmisión, el manejador enviará un carácter a la tarjeta RS-232 y se bloqueará, en espera de la interrupción generada por la interfaz al transmitir el carácter y en espera también de que

la UART pueda aceptar otro carácter. Algunas tarjetas de interfaz tienen una CPU y memoria y pueden controlar varias líneas, con lo que se encargan de gran parte del trabajo de E/S de la CPU principal.

La otra gran categoría de terminales consta de los terminales mapeados a memoria. No se comunican con la computadora sobre una línea serie, sino que son parte integral de las propias computadoras. Los terminales mapeados a memoria tienen una interfaz mediante una memoria especial llamada vídeo RAM, que forma parte del espacio de direcciones de la computadora y la CPU se dirige a ésta como al resto de la memoria.

También en la tarjeta de vídeo RAM está un chip llamado controlador de vídeo. Este chip extrae bytes del vídeo RAM y genera la señal de vídeo utilizada para manejar la pantalla (monitor). El monitor genera un rayo de electrones que recorre de forma horizontal a lo largo de la pantalla que pinta líneas en él. Por lo general, la pantalla tiene de 200 a 1200 líneas de arriba hacia abajo, con 200 a 1200 puntos por cada línea. Estos puntos se llaman píxeles. La señal del controlador de vídeo modula el rayo de electrones y determina si un píxel debe estar iluminado o no. Los monitores de color tienen tres rayos, de colores rojo, verde y azul, que se modulan en forma independiente.

Una pantalla monocromática común puede acomodar cada carácter en un cuadro de 9 píxeles de ancho por 14 píxeles de altura y tiene 25 líneas de 80 caracteres. La pantalla tendrá entonces 350 líneas de rastreo de 720 píxeles cada una. Cada uno de estos marcos se vuelve a dibujar 45 ó 70 veces por segundo. El controlador de vídeo se puede configurar de forma que busque los primeros 80 caracteres del vídeo RAM, generar 14 líneas de rastreo, buscar otros 80 caracteres en el vídeo RAM, generar las siguientes 14 líneas de rastreo, etc.

Alternativamente, un chip más barato puede evitar el almacenamiento en buffer de 80 caracteres si busca el primer carácter, lo asocia al mapa de 9 x 14 caracteres, extrae los 9 bits que necesita, los muestra y busca el siguiente carácter. Este enfoque significa que al mostrar un solo renglón de 80 caracteres, los 80 caracteres se extraen 14 veces, una vez por cada línea de rastreo. Los patrones de 9x14 bits para los caracteres se conservan en una ROM de 256 entradas que utiliza el controlador de vídeo. Su índice para una búsqueda rápida es el carácter.

Cuando la CPU escribe un carácter en vídeo RAM, éste aparece en la pantalla durante una unidad de tiempo. La CPU puede cargar una imagen de pantalla de 4K, calculada con anterioridad, al vídeo RAM en unos cuantos mseg. A 9600 bits/seg, la escritura de 2000 caracteres en un terminal RS-232 tarda 2083 mseg, que es cientos de veces más lento. Así, los terminales mapeados a memoria permiten una interacción mucho más rápida, función para la cual se utilizan.

Los terminales con mapas de bits utilizan el mismo principio, excepto que cada bit en el vídeo RAM controla de forma directa un solo píxel de la pantalla. Una pantalla de 800 x 1024 píxeles requiere 100K bytes de RAM (más si es de color), aunque permite una completa flexibilidad en los tipos y tamaños de caracteres, varias ventanas y hace posibles gráficos arbitrarios.

Con una pantalla mapeada a memoria, el teclado se desacopla totalmente de la pantalla. Tiene una interfaz por medio de un puerto paralelo, aunque también existen los teclados con interfaz RS-232. Al oprimir una tecla, la CPU tiene una interrupción y el manejador del teclado extrae el carácter oprimido al leer un puerto de E/S. A veces, las

interrupciones son generadas cuando se oprime cada tecla y también cuando se le deja de oprimir.

Además, algunos teclados sólo proporcionan el número de tecla y no el código ASCII. En el PC, por ejemplo, al oprimir la tecla A, el código de esta letra (30) se coloca en un registro de E/S. El manejador se encarga de determinar si es minúscula, mayúscula, CTRL-A, ALT-A, CTRL-ALT-A o alguna otra combinación.

6.2 Software para la entrada.

La labor básica del manejador del teclado es recolectar la entrada por medio del teclado y transferirlos a los programas. Se pueden adoptar dos filosofías para el manejador. En la primera, la labor del manejador consiste en aceptar la entrada y transferirla sin modificaciones. Esta filosofía es adecuada a las necesidades de los editores sofisticados de pantallas, que permiten que el usuario conecte una acción arbitraria con cualquier carácter o secuencia de caracteres. En la segunda filosofía el manejador controla la edición dentro de la línea y sólo entrega líneas corregidas a los programas del usuario. La primera filosofía está orientada hacia los caracteres, mientras que la segunda está orientada hacia las líneas. A menudo se conocen como modo crudo y modo cocinado, respectivamente. Muchos sistemas proporcionan ambos modos y seleccionan uno de ellos mediante una llamada al sistema.

La primera tarea del manejador del teclado es coleccionar los caracteres. Si al oprimir una tecla se provoca una interrupción, el manejador puede obtener el carácter durante tal interrupción. Si las interrupciones son convertidas en mensajes por el software de bajo nivel, es posible colocar el carácter recién obtenido en el mensaje. Otra alternativa es colocarlo en un pequeño buffer dentro de la memoria y utilizar el mensaje para indicar al manejador la llegada de algo. Este punto de vista es más seguro si sólo se puede enviar un mensaje a un proceso en espera y existe cierta probabilidad de que el manejador del teclado continúe ocupado con el carácter anterior.

Una vez que el manejador ha recibido el carácter debe empezar a procesarlo. Si el teclado entrega números de teclas en vez de códigos en ASCII, entonces el manejador debe asociar los números de las teclas con los códigos en ASCII mediante ciertas tablas.

Si el terminal utiliza el modo cocinado, los caracteres se deben almacenar hasta acumular toda una línea, puesto que el usuario podría decidir eliminar una parte de ella. Aun cuando el terminal utilizase el modo crudo, el programa podría no solicitar un dato, por lo que los caracteres deben almacenarse en un buffer para que continúe la operación de teclado.

Se utilizan dos puntos de vista para el almacenamiento de los caracteres en un buffer. En el primero, el manejador contiene un fondo central de buffers, cada uno de los cuales puede contener 16 caracteres. A cada terminal se le asocia una estructura de datos, la cual contiene, entre otros elementos, un puntero a la cadena de buffers para reunir los datos del terminal. Al escribir más caracteres, se utilizan más buffers que entran en esa cadena. Cuando los caracteres se transfieren a un programa del usuario, los buffers se eliminan y se colocan de nuevo en el fondo. El otro punto de vista consiste en hacer el almacenamiento en buffers de manera directa en la propia estructura de datos del terminal sin un fondo central de buffers.

Aunque el teclado y la pantalla son dispositivos independientes desde el punto de vista lógico, muchos usuarios están acostumbrados a ver que los caracteres recién tecleados aparezcan en la pantalla. Normalmente, el software es el encargado de mostrar

el dato. Este proceso se llama eco. El eco es complicado porque un programa puede estar escribiendo hacia la pantalla mientras el usuario esté utilizando el teclado. Al menos, el manejador del teclado debe poder decidir dónde colocar los nuevos datos sin que la salida del programa escriba sobre éstos. El eco también se complica cuando se escriben más de 80 caracteres en un terminal con líneas de 80 caracteres. Según la aplicación, podría ser adecuado pasar a la siguiente línea. Algunos manejadores sólo truncan las líneas en 80 caracteres. Otro problema es el manejo de los tabuladores. Todos los terminales tienen una tecla de tabulación, pero pocos pueden manejar los tabuladores en las salidas. El manejador se encarga de calcular la posición actual del cursor, tomando en cuenta la salida de los programas y la salida del eco y calcula el número apropiado de espacios para el eco.

Llegamos ahora al problema de la equivalencia de los dispositivos. Desde el punto de vista lógico, al final de una línea de texto uno quisiera un retorno de carro, para que el cursor regrese a la columna 1 y un alimentador de línea, para avanzar a la siguiente línea. El manejador se encarga de convertir todo lo que reciba en el formato interno estándar utilizado por el sistema operativo. Si la forma estándar consiste sólo en almacenar el alimentador de línea (la convención de UNIX), entonces los retornos de carro deben convertirse en alimentadores de línea. Si el formato interno almacena los dos, entonces el manejador debe generar un alimentador de línea al obtener un retorno de carro y un retorno de carro cuando recibe un alimentador de línea. Sin importar la convención interna, el terminal puede necesitar el eco de ambos, retorno de carro y alimentador de línea, para que la pantalla se actualice de forma adecuada. El manejador del teclado debe encargarse de convertir todas las distintas combinaciones al estándar interno del sistema y hacer que el eco se realice de la manera correcta.

6.3 Software para la salida.

El método de uso común para los terminales RS-232 es tener buffers de salida asociados a cada terminal. Los buffers pueden provenir del mismo fondo que los buffers de entrada, o bien ser exclusivos para la salida. Cuando los programas escriben hacia el terminal, la salida se copia primero en los buffers. Análogamente, la salida del eco también se copia en los buffers. Después de copiar la salida a los buffers (o cuando los buffers están totalmente ocupados), sale el primer carácter y el manejador se duerme. Cuando hay una interrupción, sale el siguiente carácter, etc.

En el caso de los terminales mapeados a memoria, los caracteres a imprimir se extraen uno a uno del espacio del usuario y se colocan de forma directa en el vídeo RAM. El manejador de un terminal mapeado a memoria debe llevar un registro en software de la posición actual del vídeo RAM, de forma que los caracteres que se puedan imprimir sean colocados en él y se avance a la siguiente posición. El retroceso, el retorno de carro y el alimentador de línea requieren que esta posición sea actualizada de forma adecuada.

Muchos de los aspectos del manejador del terminal para una pantalla mapeada a memoria (recorrido de la pantalla, timbre, etc.) también son enfrentados por el microprocesador dentro de un terminal RS-232. Desde el punto de vista del microprocesador, es el procesador principal de un sistema con una pantalla mapeada a memoria.

7 UNIX.

7.1 E/S en UNIX.

UNIX integra en el sistema de archivos lo que se conoce como archivos especiales. Cada dispositivo de E/S tiene asignado el nombre de su ruta de acceso, que, por lo general, se encuentra en */dev*. Por ejemplo, la impresora podría ser */dev/lp*.

Estos archivos especiales pueden tener un acceso igual al de otros archivos. No se necesitan comandos o llamadas especiales al sistema: las llamadas usuales al sistema, READ y WRITE se ajustan a esto. Los programas pueden abrir, leer y escribir los archivos especiales, de la misma forma que los archivos regulares. De esta forma, no se requiere de un mecanismo especial para hacer operaciones de E/S.

Una ventaja adicional es que las reglas usuales de protección de archivos se aplican de manera automática a los dispositivos de E/S. Si los bits de protección de */dev* se configuran de forma que sólo el superusuario tenga acceso directo a sus archivos, entonces los usuarios no podrán realizar la E/S por sí mismos. El acceso restringido a los dispositivos seleccionados de E/S se puede otorgar mediante la instalación de programas con permiso para leer y escribir en los archivos de */dev*, pero que hagan esto de manera limitada.

Los archivos especiales se dividen en dos categorías, de bloque y de carácter. Un archivo especial de bloque es uno que consta de una serie de bloques numerados. La propiedad fundamental de un archivo especial de bloque es que cada bloque se puede direccionar y acceder de manera individual. Los archivos especiales de bloque se utilizan para los discos.

Los archivos especiales de carácter se utilizan, por lo general, para los dispositivos de entrada o salida de un flujo de caracteres. Los terminales, impresoras, redes, ratones, plotters y la mayoría de los demás dispositivos de E/S que aceptan o producen datos para las personas utilizan archivos especiales de carácter.

Aunque los archivos especiales de carácter no pueden tener un acceso aleatorio, a menudo deben ser controlados de forma distinta a un archivo especial de bloque. Por ejemplo, consideremos un terminal. Además de aceptar la lectura y escritura de solicitudes, tiene una serie de características particulares que deben controlarse. Por ejemplo, cuando el usuario cometa un error tipográfico y desea eliminar el último carácter escrito, éste oprime cierta tecla. Algunas personas prefieren la tecla de retroceso y otras la tecla DEL. En lugar de hacer una elección y forzar a las personas a utilizarla, UNIX permite que todas estas funciones especiales y muchas otras sean adaptadas por el usuario. Se dispone de una llamada al sistema especial para configurar estas opciones. Esta llamada al sistema también controla el desarrollo de los tabuladores, permite o inhibe el eco de los caracteres, la conversión entre los retornos de carro y los alimentadores de línea y otros aspectos similares. Esta llamada al sistema no se puede utilizar en archivos regulares o archivos especiales de bloque.

7.2 Implantación de la E/S en UNIX.

La E/S en UNIX se implanta mediante una colección de manejadores de dispositivos, donde, por lo general, hay un manejador por cada dispositivo. Estos manejadores se enlazan con el sistema operativo cuando se genera el núcleo y no se pueden añadir o eliminar después de esto. Su función es la de aislar el resto del sistema de las peculiaridades del hardware. Con interfaces estándar entre los manejadores y el

resto del sistema operativo, la mayoría del sistema de E/S se puede colocar en la parte del núcleo independiente de la máquina.

El sistema de E/S se divide en dos componentes principales: el manejo de los archivos especiales de bloque y el manejo de los archivos especiales de carácter.

El objetivo de la parte del sistema que realiza la E/S en los archivos especiales de bloque (es decir, los discos) es minimizar el número de transferencias reales por realizar. Para esto, los sistemas UNIX cuentan con un buffer caché entre los manejadores del disco y el sistema de archivos. El buffer caché es una tabla en el núcleo, la cual contiene docenas o tal vez centenas de los bloques de uso reciente. Cuando se necesita un bloque de disco con cierto propósito (nodo-i, directorio o datos), primero se verifica si se encuentra en el buffer caché; en tal caso, se toma de ahí y se evita un acceso al disco.

Si el bloque no está en el buffer caché, se lee del disco al buffer y se le copia de ahí cuando sea necesario. Puesto que el buffer caché tiene espacio sólo para un número fijo de bloques, se necesita cierto algoritmo para manejarlo. Por lo general, los bloques del caché se enlazan mediante una lista enlazada. Cuando se accede a un bloque, se le desplaza a la cabeza de la lista. Si hay que eliminar un bloque del caché para hacer espacio para un nuevo bloque, se selecciona el bloque del final, puesto que éste es el bloque de uso menos reciente.

El buffer caché funciona para la escritura y la lectura. Cuando un programa escribe un bloque, se dirige hacia el caché, no hacia el disco. Sólo cuando no hay espacio en el caché y hay que reconstruir el buffer, el bloque pasa al disco. Para evitar que los bloques permanezcan por mucho tiempo en el caché antes de ser escritos al disco, todos los bloques modificados se escriben al disco cada 30 segundos.

Puesto que los archivos especiales de carácter trabajan con flujos de caracteres y no desplazan bloques de información entre la memoria y el disco, éstos no utilizan el buffer caché. En vez de esto, utilizan estructuras de datos llamadas listas-C (*C-lists*). Cada lista-C es un bloque de hasta 64 caracteres, más un contador y un puntero al siguiente bloque. Al llegar los caracteres de los terminales y otros dispositivos de carácter, éstos se almacenan en una cadena de dichos bloques.

Cuando un proceso del usuario lee de */dev/tty* (por ejemplo, la entrada estándar), los caracteres no se transfieren de manera directa de la lista-C al proceso, sino que pasan por una parte del código del núcleo llamada disciplina de línea. La disciplina de línea actúa como un filtro: toma el flujo crudo de caracteres del manejador del terminal, lo procesa y produce un flujo cocinado de caracteres. En el flujo cocinado, se ha hecho la edición local de línea, los retornos de carro están asociados con los alimentadores de línea, así como otros procesamientos especiales. El flujo cocinado pasa al proceso. Sin embargo, si el proceso desea interactuar con cada carácter, se puede poner la línea en modo crudo, en cuyo caso se evita la disciplina de línea.

La salida funciona de manera análoga: desarrolla los tabuladores como espacios, convierte los alimentadores de línea en retornos de carro + alimentadores de línea, añade caracteres de relleno después de los retornos de carro en los terminales mecánicos lentos, etc. Al igual que la entrada, la disciplina puede pasar a través de la disciplina de línea (modo cocinado) o evitarla (modo crudo). El modo crudo es particularmente enviar datos en binario a otras computadoras en una línea en serie. En este caso, no le desea conversión alguna.

8 MS-DOS.

8.1 E/S en MS-DOS.

MS-DOS soporta archivos especiales de carácter para la E/S hacia dispositivos serie, casi de la misma forma en que UNIX, excepto por el hecho de los nombres de los dispositivos no están en directorios tales como */dev*. La apertura de un archivo especial de carácter devuelve un descriptor de archivo, que se puede utilizar para la lectura y escritura.

Los archivos especiales de carácter soportan los modos crudo y cocinado, de forma similar a los archivos especiales de carácter en UNIX. En el modo cocinado, la edición entre líneas (como la eliminación de caracteres), es llevada a cabo por el sistema operativo y sólo el resultado final es disponible para el programa. En el modo crudo, los caracteres se transfieren al programa exactamente como son recibidos.

MS-DOS permite a los usuarios instalar sus propios manejadores de dispositivos adaptados justo después de arrancar el sistema. Esto difiere de UNIX, donde los manejadores de dispositivos siempre se compilan en el núcleo y no se pueden instalar después. Un manejador de dispositivo se instala al añadir un sencillo enunciado al archivo *config.sys* con el nombre de la ruta de acceso del archivo que contiene el manejador.

8.2 Implantación de la E/S en MS-DOS.

Toda la entrada y salida en MS-DOS se lleva a cabo mediante los archivos especiales de carácter y los archivos especiales de bloque. Los dispositivos de carácter manejan los dispositivos que trabajan con un carácter a la vez, como los terminales y las impresoras, mientras que los dispositivos de bloque se usan para los discos. A cada archivo especial se le asocia un manejador de dispositivo, el cual contiene el código que lleva a cabo la verdadera E/S. Algunos de los manejadores, como *com1*, *con* y *lpt1* son estándar y están dentro de *io.sys*. Los usuarios pueden cargar otros manejadores de dispositivos al arrancar el sistema a través del fichero *config.sys*.

La idea de tener manejadores de dispositivos es que MS-DOS disponga de una interfaz estándar con todos los dispositivos hardware. Cuando un programa lee o escribe en un archivo especial (incluidos todos los accesos al sistema de archivos), MS-DOS llama al manejador correspondiente de forma estándar, para indicar lo que desea que se haga, lo cual separa al sistema operativo de los detalles del hardware.

La idea de tener manejadores instalables por el usuario tiene que ver con el hecho de que las personas pueden, y de hecho lo hacen, comprar todo tipo de dispositivos especiales de E/S para sus PC. Para utilizar alguno de ellos, se debe crear un nuevo archivo especial e instalar un nuevo manejador de dispositivo para su control. En UNIX, la instalación de un nuevo manejador de dispositivo requiere volver a compilar el sistema operativo, algo poco conveniente para un producto de mercado masivo como MS-DOS.

Un manejador puede corresponder a un archivo especial de carácter o a un archivo especial de bloque, pero no a ambos, puesto que las interfaces de los dos tipos son algo distintas y las funciones que deben soportar tampoco son las mismas (por ejemplo, se permite el acceso aleatorio a los dispositivos de bloque pero no a los dispositivos de carácter).

Al instalar un manejador, se le coloca a la cabeza de una lista enlazada de manejadores. MS-DOS mantiene en una variable interna un puntero a la cabeza de la lista y lo utiliza para la búsqueda en la misma. Puesto que la lista se busca a partir de la cabeza, los manejadores de instalación más reciente tienen precedencia sobre los de instalación anterior.

Los manejadores no pueden utilizar las llamadas al sistema de MS-DOS para realizar su trabajo, puesto que MS-DOS no es reentrante, es decir, no puede aceptar una nueva llamada al sistema mientras está ocupado con una anterior.

9 Entrada/salida en Windows 2000.

9.1 Conceptos fundamentales.

El administrador de E/S colabora con el administrador Plug-and-Play. El administrador Plug-and-Play envía una solicitud a cada ranura pidiendo al dispositivo que está ahí que se identifique. Una vez que ha descubierto los dispositivos conectados, dicho administrador les asigna recursos hardware. Conforme se carga cada uno, se crea un objeto de controlador para cada uno.

El administrador de E/S también está relacionado con el administrador de consumo eléctrico. Los dispositivos de E/S pueden estar en varios estados en cuanto a consumo eléctrico. Su encendido y apagado corre por cuenta del administrador de consumo eléctrico y el administrador de E/S juntos.

La principal función del administrador de E/S es crear un marco en el que puedan operar diferentes dispositivos de E/S. La estructura básica es un conjunto de procedimientos independientes del dispositivo que se encargan de ciertos aspectos de E/S, más un conjunto de controladores de dispositivos para comunicarse con los dispositivos.

9.2 Controladores de dispositivos.

Para garantizar que los controladores de dispositivos funcionen bien, Microsoft definió un modelo de controlador al cual se espera que se ajusten los controladores de dispositivos.

Las solicitudes de E/S se pasan a los controladores en forma de paquete llamado paquete de solicitud de E/S (IRP; *I/O Request Packet*). Los controladores deben basarse en objetos en el sentido de reconocer una lista específica de métodos que puede invocar el resto del sistema. También deben interactuar con otros objetos si se les proporciona un identificador para el objeto en cuestión.

Los controladores deben manejar la función Plug-and-Play, lo que implica que si un dispositivo controlado por ese controlador se añade o quita del sistema, el controlador debe aceptar esta información y actuar en concordancia. También debe manejarse la administración de consumo eléctrico con los dispositivos para los que sea pertinente.

Los controladores deben ser configurables, lo que implica no incorporar supuestos acerca de qué líneas de interrupción o puertos de E/S usan ciertos dispositivos.

Otro requisito es poder trabajar de forma segura en multiprocesadores. Este requisito implica que mientras un controlador está procesando una solicitud a nombre de una CPU, podría llegar una segunda solicitud a nombre de una CPU distinta. El

controlador debe funcionar aunque esté siendo ejecutado al mismo tiempo por dos o más CPUs.

Para cada dispositivo administrado por un controlador se crea un objeto de dispositivo y se hace que al principio apunte al objeto de controlador. Estos objetos de controlador se colocan en un directorio especial llamado \??. Dado un objeto de dispositivo, es fácil localizar el objeto de controlador y entonces invocar sus métodos.

Un controlador puede realizar todo el trabajo él mismo, pero también es posible apilar controladores, lo que significa que una solicitud podría pasar por una serie de ellos, cada uno de los cuales realiza una parte del trabajo.

Un uso común de los controladores apilados es para separar la administración del bus y la tarea funcional de controlar en verdad el dispositivo. La administración del bus PCI es complicada y al separar esta labor de la parte específica para el dispositivo, quienes escriben los controladores tan sólo incluyen el controlador estándar de bus en su pila.

Otra razón para apilar controladores es para tener la capacidad de insertar controladores filtro en la pila. Un controlador filtro aplica alguna transformación a los datos al subir o al bajar. Por ejemplo, un controlador filtro podría comprimir datos que se envían al disco o cifrar datos que se envían a la red. Ni el programa de aplicación ni el verdadero controlador de dispositivo tienen que saber de él, y el filtro funcionará de manera automática con todos los datos que se envían al dispositivo (o llegan de él).

10 Conclusiones.

Comenzamos analizando el hardware de E/S y la relación de los dispositivos de E/S con los controladores de E/S, que son los que tienen que trabajar con el software. Después analizamos los cuatro niveles de software de E/S: las rutinas de interrupción, los manejadores de dispositivos, el software independiente de los dispositivos y las bibliotecas y spoolers de E/S que se ejecutan en el espacio del usuario. Las rutinas de interrupción guardan el estado de la máquina, después dan servicio al dispositivo y verifican los errores. En general, es recomendable utilizarlas lo menos posible, de manera consistente con el rendimiento requerido. Los manejadores de dispositivos controlan todos los detalles específicos de uno o más dispositivos. Su labor es ocultar los detalles engorrosos a los niveles superiores. El software independiente del dispositivo realiza cosas como el almacenamiento en buffers y la asignación. Este software es el mismo para varios dispositivos.

A continuación analizamos tres tipos principales de dispositivos, discos, relojes y terminales y analizamos el funcionamiento conjunto del hardware y el software. Para los discos, el algoritmo principal es la forma de planificar el movimiento del brazo. Para los relojes, lo importante es mantener un registro de varias solicitudes con un solo reloj. Para el caso de los terminales hay que distinguir entre la entrada y la salida. Los aspectos de la entrada son el modo crudo o el modo cocinado, el almacenamiento en buffers, el eco, los caracteres de relleno, la edición de líneas y el manejo de caracteres especiales. El software de salida se encarga del movimiento del cursor y otras secuencias de escape, entre otros aspectos.

En UNIX la E/S se realiza mediante los archivos especiales de carácter y de bloque, los cuales se integran al sistema de archivos. Un dispositivo de E/S de bloque utiliza un buffer caché para reducir el número de accesos al disco. Se utiliza un

algoritmo LRU para el manejo del caché. La E/S de caracteres se puede hacer en modo crudo o modo cocinado; éste último se implanta mediante una disciplina de línea.

En MS-DOS la E/S se realiza mediante archivos especiales, tanto de bloque como de carácter. La E/S se controla mediante los manejadores de dispositivos. Los usuarios pueden instalar sus propios manejadores para dispositivos especiales. Los manejadores se enlazan entre sí mediante una cadena y cada uno contiene su nombre y código.