Drivers

Los controladores de dispositivo (device drivers, en inglés) son programas se conectan con el s.o. dependiente del dispositivo. Como ejemplo tenemos:

- Interfaz SCSI.
- Norma RS232.
- Norma 802.5.
- Etcétera.

Este software, que depende del periférico en particular, soporta las particularidades de éste, y normalmente es provisto por el fabricante, aunque debe respetar los principios de compatibilidad de la norma correspondiente. Sin embargo algunos fabricantes ofrecen extensiones a las normas para diferenciar sus productos, y esto ocasiona muchas veces conflictos de compatibilidad. La mayoría de los sistemas operativos definen una interfaz estándar que todos los drivers de bloques deben soportar y una segunda interfaz estándar que todos los drivers de caracteres deben soportar. Tales interfaces consisten en varios procedimientos que el resto del sistema operativo puede invocar para pedir al driver que realice algún trabajo. Los procedimientos más comunes son los que leen y graban un bloque (dispositivo de bloques) o los que escriben y leen una cadena de caracteres (dispositivo de caracteres). En algunos sistemas, el sistema operativo es un único programa binario que contiene todos los drivers que pueda necesitar compilados dentro de él. Este esquema fue la norma durante años en los sistemas UNIX debido a que se ejecutaban en centros de cálculo en los que los dispositivos de E/S raramente cambiaban. Si se añadía un nuevo dispositivo el administrador del sistema simplemente recompilaba el núcleo con el nuevo controlador para obtener un nuevo binario. Con la llegada de los computadores personales, con la gran variedad de dispositivos de E/S, este modelo dejo de funcionar bien. Pocos usuarios son capaces de recompilar o reenlazar el núcleo incluso si cuentan con el código fuente o con los módulos objeto, lo cual no siempre es el caso. Por ese motivo, comenzando con MS-DOS los sistemas operativos cambiaron a un modelo en el que los drivers se cargan dinámicamente en el sistema durante la ejecución. Cada sistema diferente maneja la carga de los drivers de manera diferente. Un driver de dispositivo tiene varias funciones. La más obvia es la de aceptar peticiones de lectura o escritura abstractas enviadas por el software dependiente del dispositivo y controlar que se lleven a cabo, pero existen también otras funciones que debe realizar. Por ejemplo, el driver debe inicializar el dispositivo, en caso de ser necesario. También puede tener que controlar su consumo de energía eléctrica y mantener un registro de eventos.

Muchos drivers de dispositivo tienen una estructura general similar. Un driver típico comienza comprobando los parámetros de entrada para ver si son válidos. Si no lo son, se devuelve un error. Si los parámetros son válidos, puede ser necesaria una traducción de términos abstractos a concretos. En el caso de un driver de disco, esto puede significar convertir un número de bloque lineal en los números de cabeza, pista, sector correspondientes, de acuerdo con la geometría del disco. Luego el driver puede comprobar si el dispositivo está actualmente en uso. En tal caso, la petición deberá encolarse para su procesamiento posterior. Si el dispositivo está desocupado, debe examinarse el estado del hardware para ver si puede atenderse ya la petición. Puede ser necesario encender el dispositivo o poner en marcha un motor antes de comenzar las transferencias. Una vez que está encendido el dispositivo y listo para trabajar, es cuando comienza el control propiamente dicho del dispositivo. Controlar el dispositivo significa enviarle una secuencia de comandos. El driver es el lugar donde se determina la secuencia de comandos, dependiendo de la tarea a realizar. Después de que el driver determina qué comandos tiene que enviar, comienza a escribirlos en los registros de dispositivo del controlador. Después de escribir cada comando en el controlador, puede ser necesario comprobar si el controlador ha aceptado el comando y está preparado para aceptar el siguiente. Esto se repite hasta terminar de enviar toda la secuencia de comandos. Algunos controladores son capaces de leer y procesar por si mismos toda una lista enlazada de comandos (en la memoria) sin ninguna ayuda del sistema operativo posterior a la petición de procesamiento de la lista. Estos drivers se llaman "inteligentes", y el conjunto de órdenes que pueden procesar constituye lo que se

llama "protocolo". Una vez que se envían los comandos pueden darse dos situaciones. En muchos casos el driver del dispositivo debe esperar hasta que el controlador realice algún trabajo, por lo que se bloquea a si mismo hasta que llega la interrupción que lo desbloquea. Sin embargo, en otros casos, la operación termina de inmediato, por lo que el driver del dispositivo no necesita bloquearse. Un ejemplo de la segunda situación es el desplazamiento vertical (scroll) de la pantalla en modo texto, el cual sólo requiere escribir unos cuantos bytes en los registros del controlador. No se requiere ningún movimiento mecánico, por lo que toda la operación puede completarse en nanosegundos. En el primer caso, el driver bloqueado debe ser despertado por la interrupción. En el segundo caso, el driver nunca se duerme. En ambos casos, una vez que se completa la operación, el driver del dispositivo debe comprobar si se ha producido algún error. Si todo fue bien, el controlador dispone ya de los datos que debe comunicar al software independiente del dispositivo (por ejemplo, un bloque que acaba de leerse). Finalmente, el driver devuelve cierta información de estado a quien lo invocó para informarle de si todo salió bien o de si hubo errores, y cuáles fueron. Si hay más peticiones pendientes en la cola, se selecciona una de ellas y se arranca. Si por el contrario la cola está vacía, el driver se bloquea a la espera de la siguiente petición. Este sencillo modelo no es más que una burda aproximación a la realidad. Hay muchos factores que hacen que el código sea mucho más complicado. Por ejemplo, un dispositivo de E/S podría terminar mientras un driver está ejecutándose, interrumpiendo por tanto al driver. La interrupción puede provocar que se ejecute un driver de dispositivo. De hecho, podría provocar que se ejecutase nuevamente el driver actual. Por ejemplo, mientras el driver de red está procesando un paquete entrante, podría llegar otro paquete. Consecuentemente, los drivers deben ser reentrantes, lo que significa que un driver que está ejecutándose, debe tener prevista la posibilidad de que se le invoque una segunda vez antes de que haya terminado la primera llamada. En un sistema que permite la conexión en caliente es posible añadir o quitar dispositivos mientras el computador está funcionando. Como resultado, puede suceder que mientras un driver está ocupado leyendo de algún dispositivo, el sistema operativo informe de que el usuario ha quitado repentinamente el dispositivo del sistema. En ese caso no solamente es necesario abortar la transferencia de E/S actual sin dañar ninguna estructura de datos del núcleo, sino que deberán eliminarse con sumo cuidado del sistema todas las peticiones pendientes para el dispositivo ahora desaparecido, comunicando la mala noticia a quienes hicieron esas peticiones. Además, la adición inesperada de nuevos dispositivos podría obligar al núcleo a reasignar recursos (por ejemplo, líneas de solicitud de interrupción), quitándole algunos recursos al driver y dándole otros nuevos a cambio.

Los drivers a menudo necesitan interactuar con el resto del núcleo. Aunque desde los drivers no pueden realizarse llamadas al sistema, usualmente sí que se les permite que realicen llamadas a ciertos procedimientos del núcleo. Por ejemplo, usualmente hay llamadas para asignar y liberar páginas de memoria para utilizarlas como búferes. Otras llamadas útiles son necesarias para gestionar la MMU, los timers, el controlador de DMA, el controlador de interrupciones, etc. Dispositivos de bloques Las principales características de los dispositivos de bloque son:

- La información se almacena en bloques de tamaño fijo.
- Cada bloque tiene su propia dirección.
- Los tamaños más comunes de los bloques van desde los 128 bytes hasta los 1.024 bytes.
- Se puede leer o escribir en un bloque de forma independiente de los demás, en cualquier momento.
- Un ejemplo típico de dispositivos de bloque son los discos. Dispositivos de caracteres Las principales características de los dispositivos de caracter son:
- La información se transfiere como un flujo de caracteres, sin sujetarse a una estructura de bloques.
- No se pueden utilizar direcciones.
- No tienen una operación de búsqueda.
- Unos ejemplos típicos de dispositivos de caracter son las impresoras de línea, terminales, interfaces de una red, ratones, etc

Interrupciones

Una interrupción se genera cuando se quiere que la CPU deje de ejecutar el proceso en curso y ejecute una función específica de quien produce la interrupción. Cuando se ejecuta esta función específica decimos que la CPU está atendiendo la interrupción. Podemos realizar una clasificación de las interrupciones, atendiendo a la fuente que las produce. Interrupción software: se produce cuando un programa ejecuta una instrucción de interrupción INT n. En algunos sistemas operativos se deben ejecutar estas instrucciones para pedir un servicio ya sea al sistema operativo o a los programas de la BIOS. Los drivers también usan esta técnica. Se tratan de procedimientos a los que llama un programa para la ejecución de tareas ligadas, en su mayoría, al hardware de la computadora, tales como abrir un archivo, la adquisición de una tecla, enviar información a la pantalla, transmisión y recepción de datos vía un canal de comunicación con el exterior, etc. En realidad las interrupciones por software no interrumpen nada, ya que cumplen una función muy similar a una llamada a subrutina (por ejemplo, CALL en el caso de una PC). Interrupciones hardware: son causadas cuando un dispositivo hardware requiere la atención de la CPU para que se ejecute su manejador (rutina de interrupción). Internas: También llamadas excepciones. Son generadas dentro del CPU. Por ejemplo, un desbordamiento en la división, un código de operación no valido (illegal opcode), etc. Externas: Son interrupciones generadas externamente al CPU, pudiendo ser generadas por dispositivos dentro de la placa del sistema (placa madre) o por una unidad de E/S de algún periférico fuera de la placa del sistema. Por ejemplo: El teclado, Un temporizador, Un DMA al finalizar la transferencia, etc.

Estas se clasifican en:

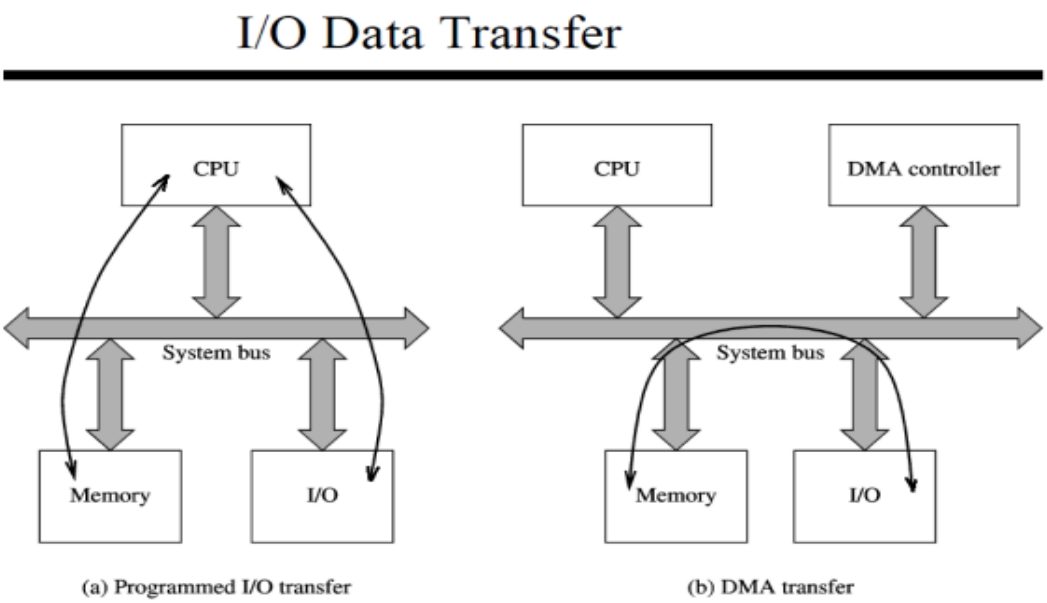
- Enmascarables: Cuando las interrupciones pueden ser activadas (permitidas) o desactivadas (inhibidas) por programa (software).
- No enmascarables: Cuando las interrupciones no pueden ser desactivadas (inhibidas) por programa. Atención de una interrupción Cuando llega un pedido de interrupción a la CPU, ya sea desde el hardware o cuando ejecuta una instrucción de interrupción, lo que se analiza para diferenciar un pedido de otro es el número de la interrupción. Este número n es provisto por la instrucción de interrupción (INT n), o proporcionado por un circuito de la CPU, denominado Controlador de Interrupciones (normalmente es un chip adicional que colabora con el microprocesador para resolver las interrupciones por hardware). Una vez que el microprocesador dispone de dicho número, el procedimiento siguiente es el mismo cualquiera fuera el origen del pedido de interrupción. Usa a dicho número como índice de una tabla en memoria conocida como Tabla de Vectores de Interrupción. Localiza a una entrada de dicha tabla donde obtiene un puntero (vector) que le permite localizar la rutina que deberá ejecutar el código que se espera para resolver el pedido de interrupción.

Acceso Directo a Memoria (DMA)

La E/S programas y la E/S por interrupciones tiene el problema de que los datos pasan por la CPU. Esto tiene los siguientes inconvenientes:

- Utiliza a la CPU para tareas rutinarias como la transferencia de bloques de bytes

- La CPU necesita muchas instrucciones (con el correspondiente acceso al BUS) por cada dato que se transfiere. Esto se soluciona con el DMA.



El acceso directo a memoria (DMA, del inglés Direct Memory Access) permite a cierto tipo de dispositivos acceder a la memoria del sistema para leer o escribir independientemente de la CPU. Muchos sistemas de hardware utilizan DMA, incluyendo controladores de unidades de disco, tarjetas gráficas y tarjetas de sonido. DMA es una característica esencial en todos los computadores modernos, ya que permite a dispositivos de diferentes velocidades comunicarse sin someter a la CPU a una carga masiva de interrupciones. Una transferencia DMA consiste principalmente en transferir un bloque de bytes entre la memoria y un dispositivo. En lugar de que la CPU controle la transferencia, la transferencia se lleva a cabo por el controlador DMA. Un ejemplo típico es mover un bloque de memoria desde una memoria secundaria externa a una interna más rápida. Tal operación no ocupa el procesador y como resultado puede ser planificado para efectuar otras tareas. Las transferencias DMA son esenciales para aumentar el rendimiento de aplicaciones que requieran muchos recursos. Cabe destacar que aunque no se necesite a la CPU para la transacción de datos, sí se necesita el bus del sistema. Por lo que existen diferentes estrategias para regular su uso, permitiendo así que no quede totalmente acaparado por el controlador DMA.

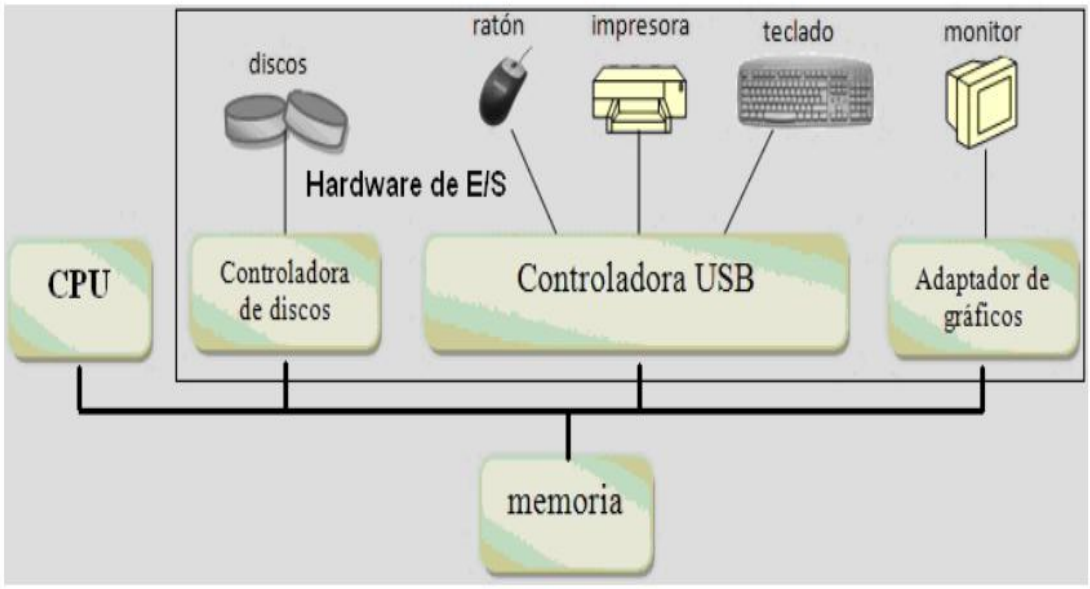
Estrategias de transferencia por DMA

A continuación se exponen diferentes técnicas para realizar la transferencia de datos. El uso de cada una de ellas dependerá de las características que se deseen primar en un sistema y de la estructura del BUS.

- DMA por robo de ciclos: permite compartir el Bus con el controlador de DMA. Se basa en usar uno o más ciclos de CPU, para el DMA, por cada instrucción que se ejecuta (de ahí el nombre). De esta forma se consigue una alta disponibilidad del bus del sistema para la CPU, aunque, en consecuencia, la transferencia de los datos será considerablemente lenta. Este método es el que se usa habitualmente ya que la interferencia con la CPU es muy baja.
- DMA por ráfagas: se dispone el BUS solamente para el DMA, durante la ejecución de este. Consiste en enviar uno o más bloques de datos solicitados, mediante una ráfaga, ocupando el bus del sistema hasta finalizar la transmisión. Así se consigue la máxima velocidad, sin embargo la CPU no podrá usar el bus durante todo ese tiempo, por lo que permanecería inactiva.
- DMA transparente: se trata de usar el bus del sistema cuando se tiene certeza de que la CPU no lo necesita, como por ejemplo en aquellas fases del proceso de ejecución de las instrucciones donde nunca se usa ya que la CPU realiza tareas internas (v. g. fase de decodificación de la instrucción). De esta manera, como su nombre indica, la DMA permanecerá transparente para la CPU y la transferencia se hará sin obstaculizar la relación CPU-bus del sistema. Como desventaja, la velocidad de transferencia es la más baja posible. • DMA dispersa: permite la transferencia de datos a varias áreas de memoria en una transacción DMA simple. Es equivalente al encadenamiento de múltiples peticiones DMA simples. De nuevo, el objetivo es liberar a la CPU de las tareas de copia de datos e interrupciones de entrada/salida múltiples.

Hardware de Entradas y Salidas

Se refiere al hardware que se conecta al sistema para ser controlado mediante los drivers del sistema operativo. Este hardware incluye a los circuitos adaptadores o controladores de periféricos como a los periféricos mismos.



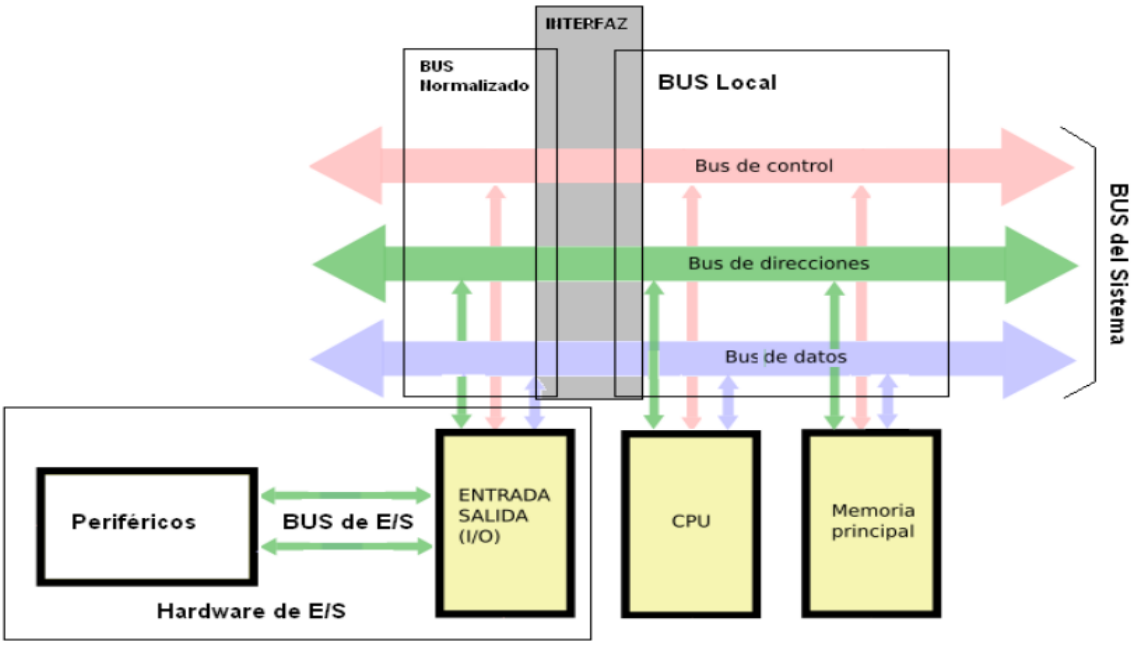
El BUS

Llamamos BUS a un conjunto de líneas de conexión, que permite vincular eléctricamente distintos módulos físicos de la computadora. En la siguiente figura podemos reconocer distintos tipos de vínculos o Buses.

El bus del sistema: es un conjunto de líneas de transmisión de bits que parten del microprocesador y llegan a los circuitos de la memoria y de E/S. Este BUS le permite al microprocesador grabar y/o leer celdas o registros en la propia CPU, en la memoria y en la E/S.

El bus local: Es el conjunto de líneas de conexión que parten directamente del microprocesador y por lo tanto trabajan a la velocidad de este y con las características físicas que impone cada microprocesador en particular.

El bus normalizado: Es un bus definido por una norma y que se conecta con el Bus Local mediante una interfaz. Existen varios tipos de ellos, como por ejemplo el BUS PCI, BUS ISA, etc. Permite desarrollar circuitos de E/S que se conectan a la computadora en forma universal, no dependiendo del microprocesador. De esta forma se puede universalizar la fabricación de placas controladoras de periféricos, como la placa de video, placa de red, etc.



El bus de direcciones: define el lugar donde se deben leer o grabar los datos. Trabaja en combinación con el bus de datos y el bus de control. La capacidad de la memoria que se puede direccionar depende de la cantidad de bits que conforman el bus de direcciones, siendo 2^n (dos elevado a la ene) el tamaño máximo en bytes del banco de memoria que se podrá direccionar con n líneas. Por ejemplo, para direccionar una memoria de 4 Gbytes, son necesarias al menos 32 líneas, pues 2^32 = 4 Gbytes. Adicionalmente pueden ser necesarias líneas de control para señalar cuando la dirección está disponible en el bus. Esto depende del diseño del propio bus.
El bus de control: permite comunicar al resto de los circuitos la actividad del microprocesador, fundamentalmente si lee o graba y en qué momento decide hacerlo. Por allí se encauzan el resto de las líneas de conexión que no conforman el bus de datos ni el bus de direcciones.

Puertos

En la forma más simple de concepción, un puerto es una memoria simple que puede almacenar un bit, un byte, un word, etc., en una operación de entrada salida. Estos puertos son celdas o registros que tienen una determinada dirección que puede estar mapeada en el espacio de direcciones físicas de la memoria principal, o tener su propio mapa de memoria. Cuando tiene su propio mapa de memoria el microprocesador deberá proporcionar una línea de control para indicar si el bus de direcciones transporta una dirección de memoria o una dirección de un puerto de entrada salida. Cuando el microprocesador ejecuta una instrucción de INPUT u OUTPUT, envía por el bus de direcciones el código de la dirección de un puerto de entrada o salida, y por el bus de control indica mediante una señal de un bit, que dicha dirección se refiere a una entrada salida y no a la memoria. En nuestra PC cada dispositivo tiene un conjunto de direcciones de entrada salida que comúnmente llamamos puertos de entrada salida. Estas direcciones están normalizadas de tal manera que cada tipo de dispositivo tiene un rango de direcciones reservadas. Mediante instrucciones de entrada (input) o salida (output) se pueden leer o grabar datos o cualquier tipo de señal digital en cualquier dispositivo físico aparte de lo que es la memoria principal, que posea memorias o celdas (también le llamamos puertos, para reconocer que se trata de un dispositivo o circuito periférico). Estos dispositivos que poseen puertos pueden ser:

- Circuitos secuenciales con memoria (flip-flops)
- Circuitos integrados de mediana y gran escala de integración que posen registros (puertos) como los controladores de DMA, controladores de interrupciones, microcontroladores de dispositivos.
- Placas adaptadoras de periféricos que posen los componentes anteriores