1 S

1.1 Sumario

2 Funciones y Objetivos

2.1 Introducción

Como ya hemos visto en repetidas ocasiones, los 2 objetivos principales de un Sistema Operativo son, la Interfaz de Máquina Extendida (abstracción de los elementos físicos y dependientes del hardware en modelos más comunes y manejables) y la Gestión de Recursos. Toda computadora debe actuar con su entorno, bien recibiendo información de Entrada, o bien mostrando información de Salida; resultado de aplicar un programa a un conjunto de datos de Entrada. Por tanto, la gestión de E/S es fundamental en la operativa de un sistema. Por dispositivos de E/S hacemos referencia a todos los dispositivos que interactuan con el exterior, es decir aquellos que actúan como periféricos y que se pueden conectar a la computadora. La conexión puede realizarse bien explícitamente, o bien formando parte del propio hardware de la misma. El Sistema Operativo debe comunicarse con ellos y, por tanto, disponder de todos los mecanismos para llevar a cabo la comunicación, siendo el encargado de controlar el acceso de los procesos a los mismos.

2.2 Dispositivos

Los dispositivos son los componentes físicos cuya labor es realizar algún tipo de transferencia desde y/o hacia la CPU. Consta de 2 partes físicas:

- **El dispositivo**: En sí, es decir el propio dispositivo físico

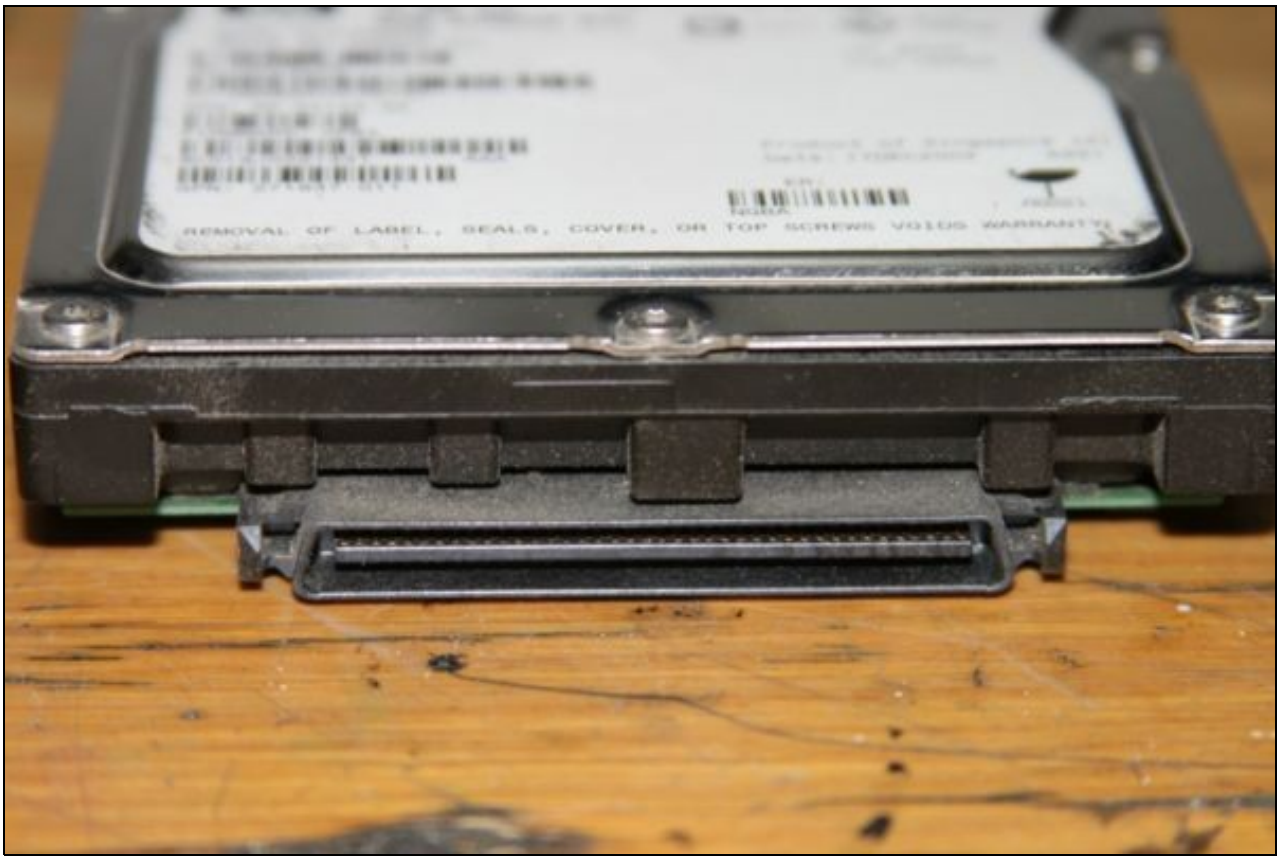


Imagen de un disco de tipo SCSI. Todos los discos de este tipo, independientemente del fabricante, se conectan de un mismo modo a la placa, a través de una controladora SCSI en ésta. En la imagen puede verse claramente el conector SCSI que permitirá conectar el disco a la controladora en la placa base.

- **Adaptador o Controladora Hardware**: Es la parte de la electrónica del dispositivo que adapta o estandariza su funcionamiento para la interconexión con hardware genérico.

Por ejemplo: Un disco duro, de un fabricante X, dispone de unas características física y constructivas concretas, diferentes de las de un dispositivo del fabricante Y. Sin embargo, a la hora de conectar esos dispositivos a una placa, para el Sistema Operativo éstos deben de ser vistos de un modo uniforme, sin grandes diferencias entre ellos. Este trabajo es el que realiza el Adaptador o Controladora, el cual adapta e independiza el dispositivo del hardware real que define su funcionamiento. Concretando un poco más, existen especificaciones y estándares de la industria que definen interfaces de conexión o adaptación de discos duros. Algunos de ellos, muy conocidos, son: IDE, SATA, SAAS, SCSI. Todos ellos vienen definidos por un estándar internacional, que unifica su estructura y el modo de comunicarse con ellos. La electrónica responsable de implementar esas características estándar es la que constituye la Controladora del dispositivo. Este principio rige para todos los dispositivos, susceptibles de conectarse de un modo, o mediante una interfaz concreta, por ejemplo: USB, FireWire, SATA, IDE, etc. Los dispositivos derivados y que siguen la especificación correspondiente, un disco externo USB, un disco interno SCSI, un disco SATA, un Pendrive, una impresora, etc., deberán de incorporar en su electrónica, o al menos disponer del conector de adaptación correspondiente en caso de que esa electrónica esté en la placa base, del interfaz correspondiente para trabajar con la especificación concreta. Es muy importante **no confundir**, por un lado la **Controladora o Adaptador Hardware**, y el **controlador de dispositivo software (driver)**. Aunque para ambos se utiliza asiduamente el término "controlador/a" el primero es físico (electrónico) y el segundo es un componente software que se integra con el Sistema Operativo para comunicarse con el correspondiente dispositivo.

En esta imagen puede verse una controladora SATA, a la que se pueden conectar dispositivos (discos, unidades ópticas) que se adapten a esa especificación de interfaz

Los dispositivos se clasifican en función de:

- Tipo de interacción con el sistema
    - Entrada: Teclado, ratón, micrófono...
    - Salida: Pantalla, impresora, altavoces...
    - Entrada/Salida: Pantalla táctil, impresora multifunción (con escáner), etc.

- Modo de funcionamiento
    - Bloques: Funcionan intercambiando información haciendo referencia (direccionando) unidades de almacenamiento, denominadas bloques. Por ejemplo: disco duro, CD-DVD...
    - Caracteres: Funcionan intercambiando información mediante una secuencia de símbolos ordenados. Por ejemplo: módem, tarjeta de red, pantalla, teclado, ratón, impresora...

2.3 Objetivos

Ya estamos en condiciones de enunciar los objetivos y funcionalidad que debe de proporcionar un Sistema Operativo en lo relativo a la gestión de la E/S:

- **Independencia del dispositivo**: Gestionar los dispositivos de un modo abstracto, de modo que éstos se vean como elementos de E/S sin más. De este modo las aplicaciones, a través del Sistema Operativo, pueden comunicarse con los dispositivos de un modo sencillo y normalizado. Por ejemplo con llamadas como: "Abrir dispositivo", "Leer de dispositivo", "Escribir en dispositivo", etc., realizadas lógicamente a través de las interfaces de librerías de los lenguajes de programación correspondientes.
- **Denominación uniforme**: Utilizar un modelo de denominación coherente y compartido por cualquier dispositivo de E/S. El caso más importante es el sistema de denominación de archivos y directorios, que utiliza conceptos como ruta relativa y ruta absoluta para hacer referencia a éstos.
- **Manejo de errores**: Ocultar a los programas las condiciones de error que hayan podido ocurrir durante el acceso a un dispositivo de E/S, de modo que el usuario no sea consciente de los reintentos en caso de fallo de una operación. En caso de que una operación no pueda realizarse se notificará simplemente con un mensaje de error.
- **Almacenamiento intermedio de datos (búffer)**: Todos los datos transferidos a y desde un dispositivo a la memoria principal deben de almacenarse provisionalmente en zonas de memoria intermedia, o búffers. El motivo de que esto sea así es principalmente por las diferencias de velocidad de operación de CPU, Memoria y dispositivos y a la ejecución concurrente de procesos (multitarea), no puede saberse a priori cuando realmente se hará una transferencia y cuánto tiempo tardará. Por este motivo se reservan zonas de memoria intermedia en el propio dispositivo (en la controladora hardware) y en la memoria principal de la computadora.

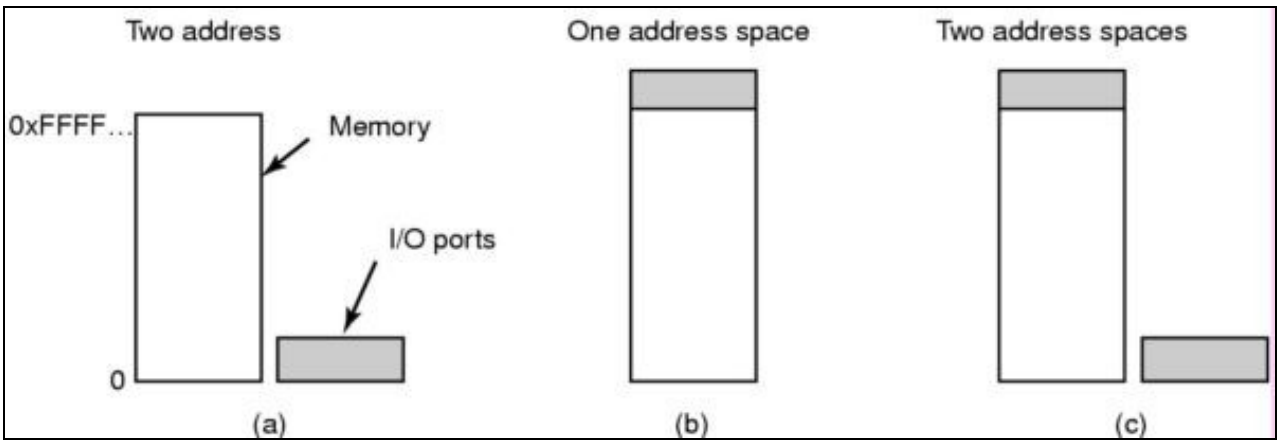
3 Espacios de direccionamiento

La forma que tiene el Sistema Operativo de conocer el estado de un dispositivo es leyendo la información que el propio dispositivo le proporciona. Por tanto, para hacer referencia a un dispositivo necesitamos un modo de poder comunicarnos con él. La manera de controlar de este aspecto suele venir

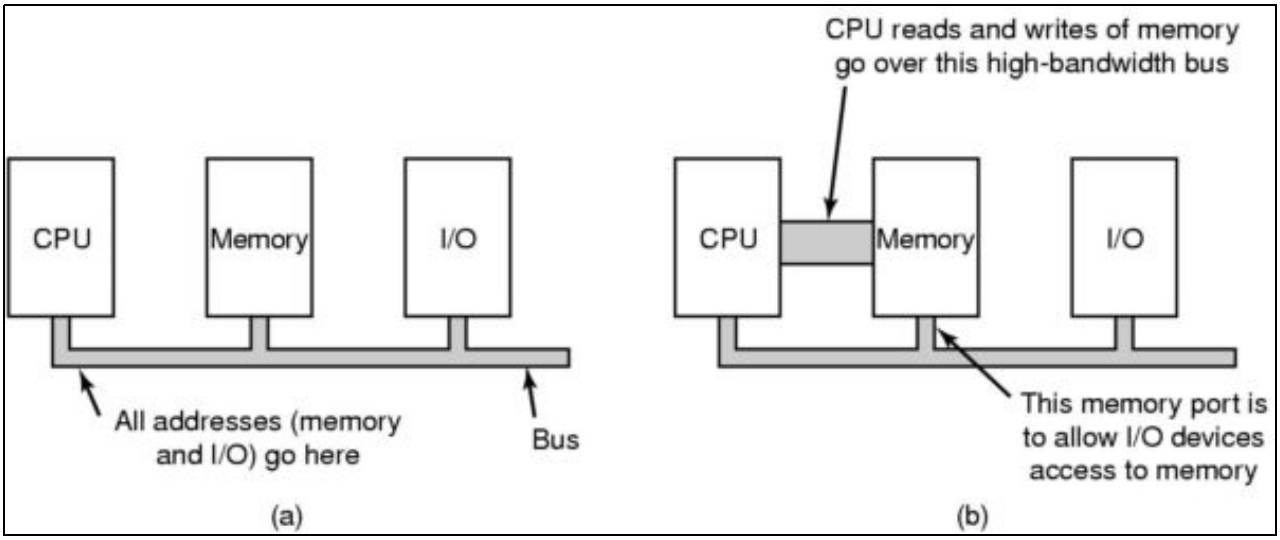
parcialmente integrada en la propia lógica de la controladora hardware. La controladora hardware dispone de una serie de registros de estado y de un búffer de datos que deben ser accesibles de algún modo para la CPU. La clave está en cómo puede la CPU visualizar estos registros imprescindibles para comunicarse con el dispositivo. Existen 2 alternativas:

- **Espacios de direcciones separados para memoria y dispositivos**: En este caso la CPU utiliza 2 modos de direccionamiento independientes, uno para identificar los registros de los dispositivos en sí, que se denominan **Puertos de E/S** y otro para hacer referencia a la propia memoria principal. La CPU dispone de instrucciones diferentes para realizar operaciones del tipo "acceso a Puerto de E/S" y otras instrucciones para operaciones del tipo "Acceso a memoria principal".

- **Único espacio de direcciones para memoria y dispositivos**: En esta modalidad se utiliza un direccionamiento uniforme para dispositivos y para la memoria principal, de modo que, para la CPU no hay diferencias en este aspecto. Por tanto las instrucciones de acceso a datos en la memoria principal y a la información de un dispositivo son las mismas. En sistemas que utilizan esta aproximación, la mayoría en la actualidad, se reserva una zona de la memoria principal para ubicar los Puertos de E/S de los dispositivos, habitualmente un espacio reservado del orden de los KB, y el resto del espacio de direcciones de la memoria principal se utiliza para la asignación de memoria a los espacios de direcciones de los procesos.



a) Espacios separados para Puertos de E/S y memoria
b) Espacio único para memoria y Puertos de E/S
c) Modelo mixto



a) Canal compartido para direcciones referentes a memoria y puertos de dispositivos
b) Canales físicos separados para direcciones de memoria y puertos de dispositivos

Las ventajas del segundo modelo en relación al primero son:

- No se necesitan instrucciones específicas de la CPU para el acceso a los dispositivos
- Se pueden construir el software que accede a los dispositivos utilizando un lenguaje de programación de alto nivel, como C. En la aproximación basada en separar Puertos de E/S y memoria principal es necesario escribir las instrucciones de acceso a Puertos de E/S en lenguaje ensamblador

4 Modalidades de E/S

4.1 E/S programada

El sistema más simple para gestionar la E/S es dejarle todo el trabajo a la CPU. En la modalidad de E/S programada las transferencias de datos de E/S son supervisadas y atentidas por la CPU, de modo que mientras realiza este trabajo no puede atender ninguna otra tarea del sistema. Aún así la E/S programada se utiliza en casos en los que la complejidad de una alternativa no compensa la simplicidad inherente a la E/S programada. Un ejemplo es la gestión de dispositivos de bajo tráfico y operación asíncrona, este es el caso por ejemplo de teclado y ratón. La CPU "sondea" periódicamente estos dispositivos para determinar si han enviado datos desde el último sondeo. Este modo de operación se denomina "polling" (sondeo) y solo se utiliza en casos concretos como el anterior. Para dispositivos que tienen que enviar cantidades de información grandes (como discos, unidades ópticas, conexiones de red, etc.) no es adecuado el uso de la E/S programada.

4.2 E/S controlada por interrupciones

Una evolución conceptual para gestionar la E/S es el punto de vista basado en interrupciones. En esta modalidad, en lugar de que la CPU tenga que sondear al dispositivo en cuestión (accediendo a la información en los registros de su controladora) es el propio dispositivo el que notifica a la CPU por cada operación de transferencia realizada. Aún así la transferencia de datos del dispositivo a memoria principal sigue estando controlada por la CPU, sin embargo durante el tiempo en que se leen los datos del dispositivo, o se escriben en éste en caso de transferencias de salida, éste no molesta de ningún modo a la CPU que podrá atender a otros procesos del sistema.

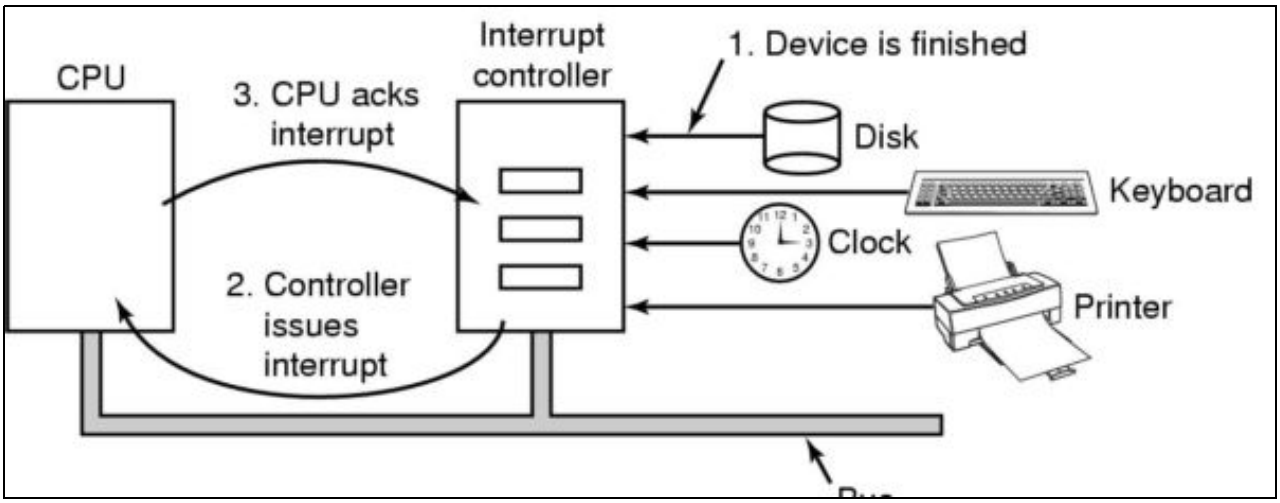


Gráfico que ilustra la operativa de un procedimiento de interrupción

4.3 E/S mediante DMA

Este es el modelo de E/S más sofisticado y eficiente, sobretodo para dispositivos con una alta demanda de transferencia de datos (discos, cd, dvd, conexiones de red, etc.). DMA solventa las carencias del modelo basado en interrupciones, el cual no es eficiente. El motivo de esto es que, en el modelo basado en interrupciones, para cada transferencia desde el búffer de la controladora del dispositivo a memoria principal, la CPU tiene que intervenir para escribir esos datos en memoria. Lo mismo para el caso de escrituras, la CPU debe tomar el control para transferir los datos desde la memoria principal al búffer del dispositivo. Para solucionar este problema, desde hace años, se viene utilizando el sistema DMA (Directory Memory Access, Aceso Directo a Memoria). Con la ayuda de un componente hardware, **la controladora DMA**, la cual suele venir integrada en la placa para poder ser utilizada por varios dispositivos, se "delegan" las operaciones de E/S a la DMA, de modo que mientras dure la transferencia completa de datos, a o desde el dispositivo, la CPU no tenga que intervenir en absoluto, quedando liberada por completo para atender otros procesos durante las transferencias de E/S.

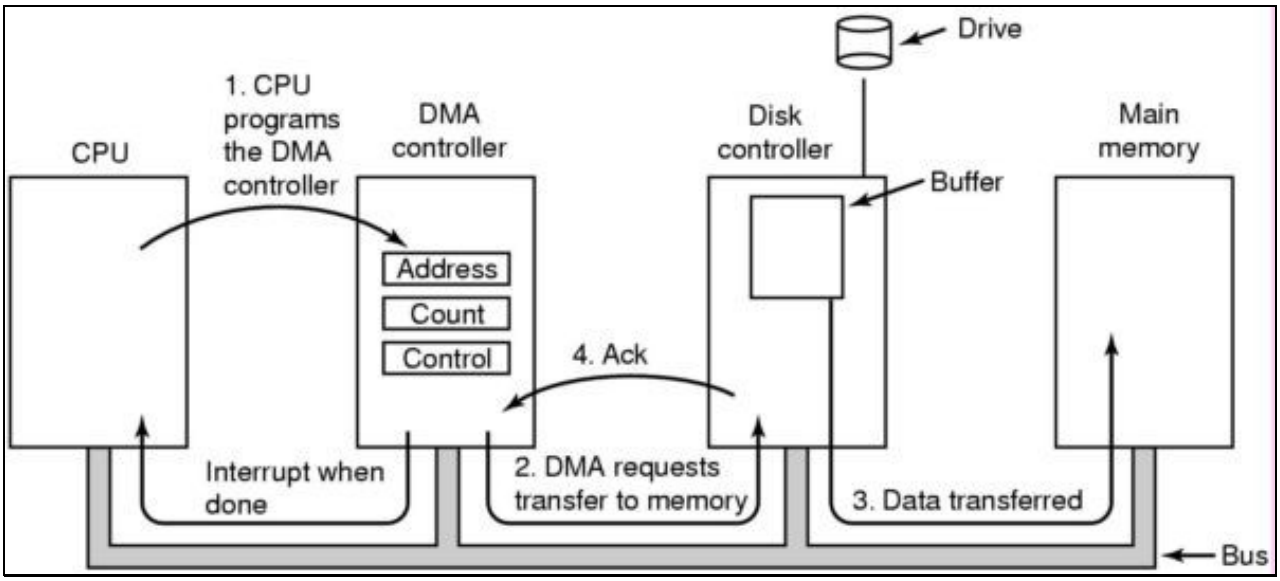


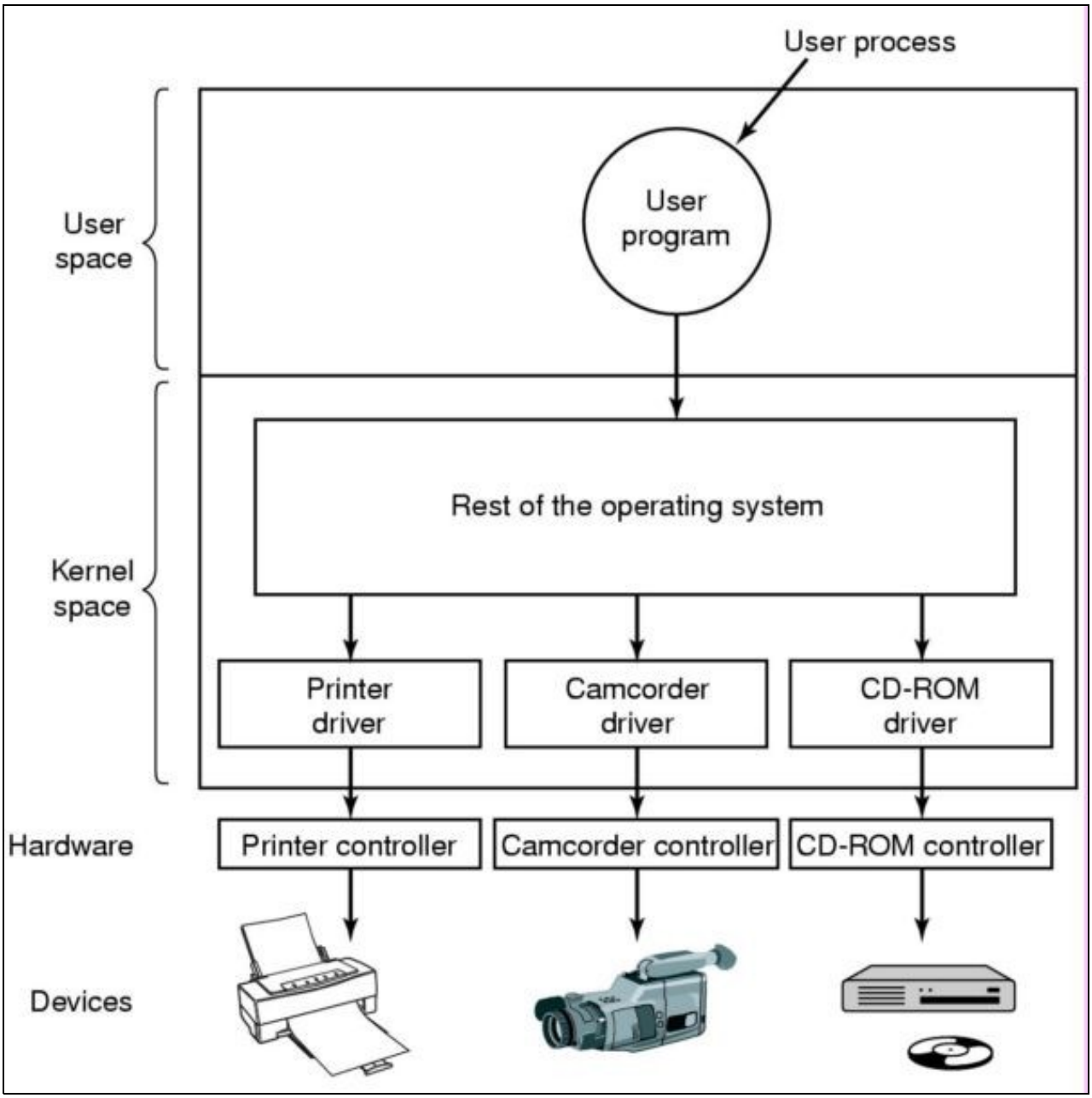
Gráfico que muestra la operativa relacionada con una transferencia de E/S controlada por DMA

1. La CPU programa la controladora DMA para realizar la transferencia completa, indicando la dirección de memoria a partir de la cual se ubicarán los datos leídos y el número de transferencias a realizar
2. La DMA, acepta la petición y envía la primera petición de lectura del primer bloque de datos a la controladora del disco
3. Tras armar el búfer de datos, la controladora transfiere a memoria el bloque actual
4. La controladora de disco envía un acuse, ACK, de lectura del bloque a la DMA
5. El ciclo 2, 3, 4 se repite hasta que se lean todos los bloques solicitados inicialmente a la controladora DMA por la CPU
6. Cuando terminan de transferirse a memoria todos los bloques la controladora DMA interrumpe a la CPU para indicar que la transferencia completa ha finalizado

5 Capas o Niveles de E/S

5.1 Introducción

La gestión de E/S es una labor complicada y extensa debido al ecosistema de dispositivos, interfaces y estándares de interconexión. Por este motivo, siguiendo el mismo principio que en el diseño de arquitectura de redes de computadoras, se utiliza para definir su estructura un modelo basado en capas o niveles. Los modelos en capas permiten dividir un problema en aspectos, desde lo más concreto a lo más abstracto, utilizando una visión top-down (arriba-abajo) para diseñar un sistema de gestión adecuado y flexible.



En este gráfico puede observarse la gestión de E/S del Sistema Operativo en un modelo basado en capas, desde lo más abstracto (capa superior) que representa el punto de vista del usuario, hasta lo más concreto (capa inferior) dónde se pueden ver los distintos dispositivos hardware. En los Sistemas Operativos actuales se utiliza un modelo en capas que responde a la estructura expuesta a continuación, desde lo más concreto y detallado (rutinas de tratamiento de interrupciones), hasta lo más general y abstracto (E/S en espacio de usuario)

5.2 Rutinas o procedimiento de tratamiento de interrupciones

En el nivel más bajo el Sistema Operativo tiene que encargarse de los sucesos asíncronos que proceden de los dispositivos. Es decir, un dispositivo realiza los trabajos encomendados, pero no es posible anticipar cuando terminará de hacer esa tarea, por ese motivo es un modelo de naturaleza asíncrona. Las interrupciones son eventos o notificaciones que envían los dispositivos, a través de su controladora hardware, y que deben ser atendidas con urgencia por la CPU. Asociadas al tratamiento de las interrupciones están las rutinas o procedimientos de tratamiento de interrupción, los cuales realizan las tareas encomendadas para la atención y necesidades del dispositivo que provocó la interrupción. Por tanto, es mediante la interrupción y las rutinas o procedimientos de servicio a la interrupción, que se define la barrera que separa a hardware de software, constituyendo las fronteras de las mismas del lado del hardware las interrupciones y las rutinas de servicio a la interrupción desde el punto de vista del software. Éstos son aspectos de muy bajo nivel, relacionados directamente con el hardware y por eso se ubican en la primera capa del software de E/S del Sistema Operativo.

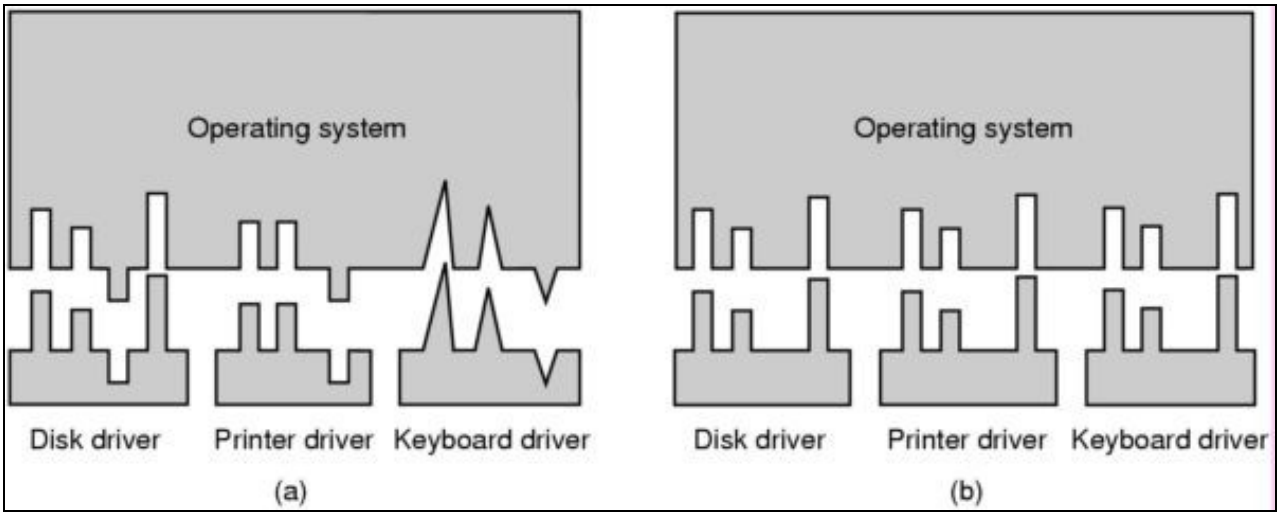
5.3 Controladores de dispositivo o Drivers

Los Controladores de dispositivo o Drivers son componentes software especializados en comunicarse con un determinado dispositivo. En general son proporcionados por los fabricantes del hardware, que conocen los detalles específicos de los dispositivos que fabrican. El Sistema Operativo se comunica con ellos utilizando un lenguaje "estándar", independiente del dispositivo, y es el propio Driver el que interpreta "a su manera" las órdenes que recibe del Sistema Operativo para traducirlas al lenguaje de la Controladora hardware que se comunica con el propio dispositivo. Por cuestiones de rendimiento y eficiencia suelen ejecutarse en modo privilegiado (modo kernel) y por tanto son componentes software críticos para el Sistema Operativo. Un fallo de programación o un error en un Driver puede detener o dañar gravemente el estado de ejecución de un sistema. Por este motivo suelen supervisarse estrechamente por el Sistema Operativo las tareas de implantación de nuevos Controladores, de manera que éstos deben de verificar una serie de condiciones y aspectos de diseño concretos.

5.4 E/S independiente del dispositivo

El "lenguaje estándar" al que se hace referencia en el apartado anterior se refiere al modo en el que el propio Sistema Operativo interactúa con los Drivers a petición de los programas de aplicación. Esta pieza fundamental que interconecta esos dos aspectos es el software de E/S independiente del dispositivo. Los aspectos controlados en este nivel son los siguientes:

- **Interfaz uniforme con los drivers de los dispositivos**: El modo en que el Sistema Operativo se comunica con los Drivers debe ser lo más uniforme y genérico posible, de modo que sea fácil comunicarse con los dispositivos de un modo "universal". Para ello se estandariza el mecanismo de comunicación (protocolo) de modo que se pueda añadir nuevos drivers sin necesidad de cambiar la forma en la que el Sistema Operativo dialoga con ellos.



En este gráfico se muestra, en la parte derecha, como mediante una estructura, interfaz, de comunicación uniforme es más sencillo integrar los controladores de dispositivo (drivers) en el Sistema Operativo, mejorando la estabilidad y flexibilidad del mismo

- **Gestión de los búffers**: Otra de las tareas de esta capa es gestionar los búffers, o memorias intermedias, en la memoria principal que se encargan de acumular los datos leídos (o que se van a escribir) en un dispositivo.

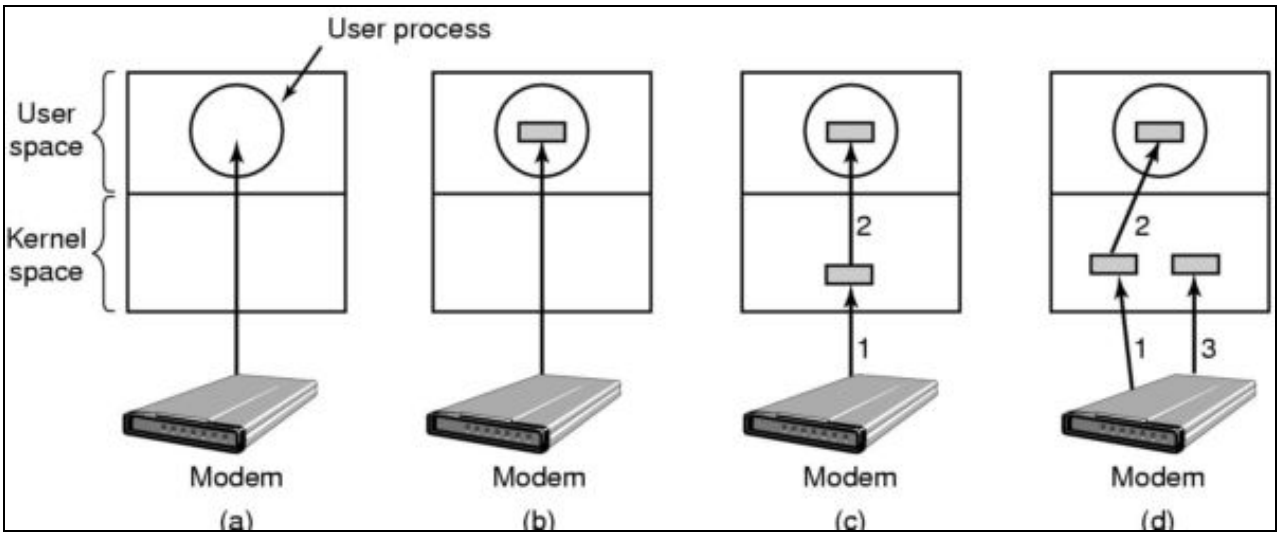


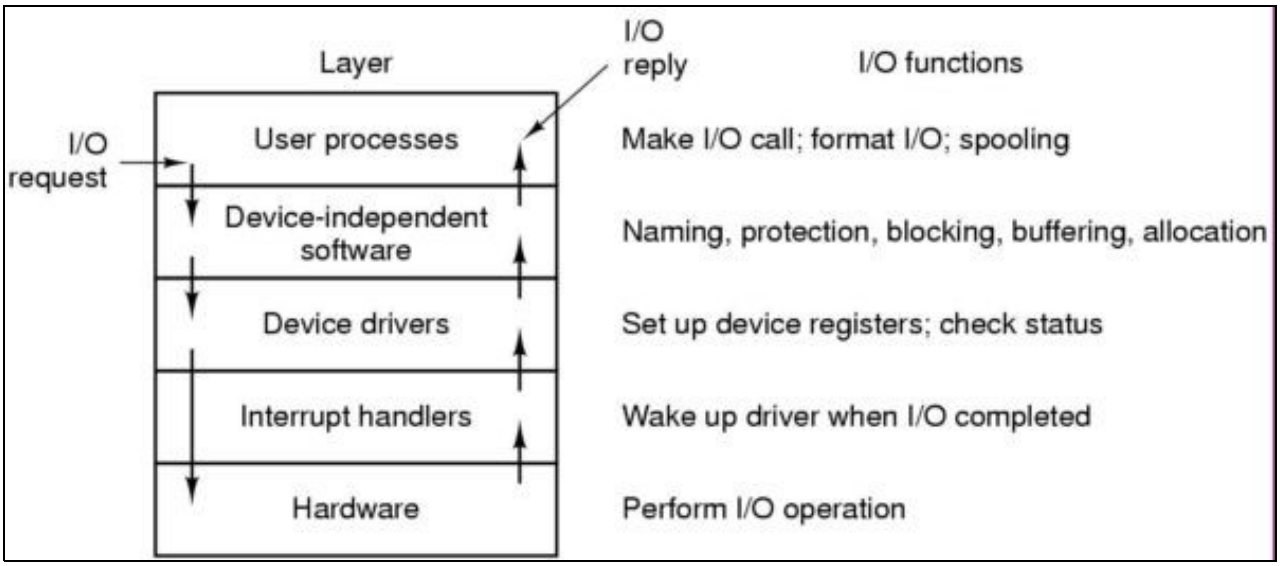
Gráfico ilustrativa del uso de búffers de lectura en la memoria principal. Estos búffers almacenan los datos que se van leyendo del dispositivo. Vemos en la figura que se están utilizando búffers en espacio de kernel y espacio de usuario. Las lecturas se hacen en primer lugar al espacio de kernel, modo en el cual se puede realizar el acceso al dispositivo, y luego se copian al espacio de usuario, donde las aplicaciones pueden leer sus datos. En la figura de la derecha se ilustra la técnica de doble búffer, el cual se utiliza para que mientras se copian datos de búffer de kernel al del usuario puedan seguir recibiéndose datos del dispositivo, para ello se utiliza otro búffer del kernel.

- **Informar de los errores**: El reporte de errores de las capas bajas se intercepta en la capa de E/S independiente del dispositivo dónde serán tratados.
- **Asignación y liberación de dispositivos dedicados**: Los procesos en ocasiones necesitan un acceso exclusivo a los dispositivos. Esta capa se encarga de gestionar este acceso, asignando y liberando los dispositivos a los procesos
- **Proporcionar un tamaño de bloque independiente del dispositivo**: El tamaño de bloque físico de los dispositivos de tipo bloque, como los discos, está predefinido en el hardware. Sin embargo para el Sistema Operativo suele ser más adecuado trabajar con tamaños de bloque que son múltiplos del tamaño de bloque físico (usualmente 512 bytes), de modo que las lecturas de bloques desde el punto de vista del Sistema Operativo (bloques lógicos) en realidad se traducen en órdenes de lectura de varios bloques físicos para el dispositivo. Por la propia naturaleza de los discos y su funcionamiento suele ser mucho más eficiente leer o escribir varios bloques físicos en cada operación de E/S, según el tamaño de bloque lógico, que hacerlo directamente leyendo o escribiendo en unidades de bloque físico (512 bytes). Esto se debe a que los retrasos de posicionamiento de la cabeza de lectura/escritura del disco son mucho mayores que los tiempos de transferencia de los bloques, por tanto, es más eficiente leer varios bloques físicos a la vez que hacerlo de uno en uno.

Por ejemplo: Si tenemos bloques lógicos de 4KB, cada lectura de un bloque lógico de disco corresponderá con 8 lecturas de bloque físico (512 bytes x 8 = 4KB) Además esto permite que se puedan definir tamaños de bloque lógico diferentes (4KB, 8KB, 16KB), lo cual es útil para optimizar el rendimiento en función del tipo de utilización que el Sistema Operativo hará de los datos en el dispositivo (por ejemplo, para sistemas que manejen archivos grandes es más adecuado el uso de un bloque lógico grande)

5.5 E/S en espacio de usuario

Y para terminar, en la parte más alta (abstracta) del modelo de capas nos encontramos con la E/S en espacio de usuario. Esta es la capa correspondiente al punto de vista que las aplicaciones y usuarios tienen del sistema de E/S. Por tanto consistirá en una serie de librerías (API, Application Programming Interface) que definen el modo para acceder a los dispositivos desde un punto de vista independiente de dispositivos. Por tanto, a la hora de utilizar un programa o escribir un programa de aplicación, el usuario no debe de preocuparse de los detalles del dispositivo, ni de aspectos de denominación del mismo, ni de manejo de los búffers, ni del reporte de errores, etc., ya que éstos son aspectos gestionados por la capa inmediatamente inferior (E/S independiente del dispositivo) que a su vez se comunica con la capa de Drivers de dispositivo. Ejemplo: A la hora de mandar un documento a la impresora simplemente invocamos esa función en un programa o en una librería de funciones, si estamos programando. No somos realmente conscientes de que existe un proceso, dentro del E/S en espacio de usuario, que se encarga de gestionar realmente la asignación de la impresora a los procesos que la necesitan. Este proceso es el servicio o demonio de spool, que gestiona la cola de documentos de la impresora. Este es otro ejemplo de lo que esta capa realiza, poniendo a disposición de usuarios y programadores un modelo cómodo y sencillo de utilizar los dispositivos de E/S



En resumen, en este gráfico vemos las capas del sistema de E/S y su interacción

6 Referencias

6.1 Búfer Caché

http://www.tldp.org/pub/Linux/docs/ldp-archived/system-admin-guide/translations/es/html/ch07s06.html

Volver

JavierFP 13:10 04 dec 2018 (CET)