Entrada/Salida

JOAQUÍN ROIZ PAGADOR[1]

[1] *Universidad Pablo de Olavide, Ctra. Utrera 1, 41013, Dos Hermanas, Sevilla, España.*
[*] *Contacto: quiniroiz@gmail.com*

Compiled June 1, 2018

Resumen del temario de la asignatura de Sistemas Operativos del Grado en Ingeniería Informática en Sistemas de Información.

OCIS codes: (130.6750) Systems.

quiniroiz@gmail.com

1. PRINCIPIOS DEL HARDWARE DE E/S

Los programadores ven la interfaz que se presenta al software: los comandos que aceptan el hardware, las funciones que lleva a cabo y los errores que se pueden reportar.

A. Dispositivos de E/S

Los dispositivos de E/S se pueden dividir básicamente en dos categorías: **dispositivos de bloque** y **dispositivos de carácter**. Un dispositivo de bloque almacena información en bloques de tamaño fijo, cada uno con su propia dirección. Los tamaños de bloques comunes varían desde 512 bytes hasta 32,768 bytes.

Se puede concluir que no está bien definido el límite entre los dispositivos que pueden direccionarse por bloques y los que no se pueden direccionar así.

Un dispositivo de carácter envía o acepta un flujo de caracteres, sin importar la estructura del bloque. No es direccionable y no tiene ninguna operación de búsqueda.

Este esquema de clasificación no es perfecto. Algunos dispositivos simplemente no se adaptan. La mayoría de estos dispositivos tienden a hacerse más rápidos a medida que pasa el tiempo.

Dispositivo	Velocidad de transferencia de datos
Teclado	10 bytes/seg
Ratón	100 bytes/seg
Módem de 56K	7 KB/seg
Escáner	400 KB/seg
Cámara de video digital	3.5 MB/seg
802.11g inalámbrico	6.75 MB/seg
CD-ROM de 52X	7.8 MB/seg
Fast Ethernet	12.5 MB/seg
Tarjeta Compact Flash	40 MB/seg
FireWire (IEEE 1394)	50 MB/seg
USB 2.0	60 MB/seg
Red SONET OC-12	78 MB/seg
Disco SCSI Ultra 2	80 MB/seg
Gigabit Ethernet	125 MB/seg
Unidad de disco SATA	300 MB/seg
Cinta de Ultrium	320 MB/seg
Bus PCI	528 MB/seg

Fig. 1. Velocidades de transferencia de datos comunes de algunos dispositivos, redes y buses

B. Controladores de dispositivos

Las unidades de E/S consisten en un componente mecánico y un componente electrónico. EL componente electrónico se llama **controlador de dispositivo** o **adaptador**. En las computadoras personales, comúnmente tiene forma de un chip en la tarjeta principal o una tarjeta de circuito integrado que se puede insertar en una ranura de expansión (PCI). El componente mecánico es el dispositivo en sí.

La interfaz entre el controlador y el dispositivo es a menudo de muy bajo nivel. Lo que en realidad sale del disco es un flujo serial de bits, empezando con un **preámbulo**, después los 4096 bits en un sector y por último una suma de comprobación, también conocida como **Código de Corrección de Errores (ECC)**.

2. FUNDAMENTOS DEL SOFTWARE DE E/S

A. Objetivos del software de E/S

Un concepto clave en el diseño del software de E/S se conoce como **independencia de dispositivos**. Lo que significa es que debe ser posible escribir programas que puedan acceder

a cualquier dispositivo de E/S sin tener que especificar el dispositivo por adelantado.

Un objetivo muy relacionado con la independencia de los dispositivos es la **denominación uniforme**. El nombre de un archivo o dispositivo simplemente debe ser una cadena o un entero sin depender del dispositivo de ninguna forma.

Otra cuestión importante relacionada con el software de E/S es el **manejo de errores**. En general, los errores se deben manejar lo más cerca del hardware que sea posible. Si el controlador descubre un error de lectura, debe tratar de corregir el error por sí mismo. Si no puede, entonces el software controlador del dispositivo debe manejarlo, tal vez con sólo tratar de leer el bloque de nuevo.

Otra cuestión clave es la de las transferencias **síncronas** contra las **asíncronas**. La mayoría de las operaciones de E/S son asíncronas: la CPU inicia la transferencia y se va a hacer algo más hasta que llega la interrupción. Los programas de usuario son mucho más fáciles de escribir si las operaciones de E/S son de bloqueo: después de una llamada al sistema read, el programa se suspende de manera automática hasta que haya datos disponibles en el búfer. Depende del sistema operativo hacer que las operaciones que en realidad son controladas por interrupciones parezcan de bloqueo para los programas de usuario.

Otra cuestión relacionada con el software de E/S es el **uso de búfer**. A menudo los datos que provienen de un dispositivo no se pueden almacenar directamente en su destino final. Cuando un paquete llega de la red, el sistema operativo no sabe dónde colocarlo hasta que ha almacenado el paquete en alguna parte y lo examina.

B. E/S programada

Es más simple de ilustrar la E/S programada. Primero ensambla la cadena en un búfer en espacio de usuario. Después el proceso de usuario adquiere la impresora para escribir, haciendo una llamada al sistema para abrirla. Si la impresora está actualmente siendo utilizada por otro proceso, esta llamada fallará y devolverá un código de error o se bloqueará hasta que la impresora esté disponible, dependiendo del sistema operativo y los parámetros de la llamada. Una vez que obtiene la impresora, el proceso de usuario hace una llamada al sistema para indicar al sistema operativo que imprima la cadena en la impresora.

Después, el sistema operativo por lo general copia el búfer con la cadena a un arreglo, por ejemplo, p en espacio de kernel, donde se puede utilizar con más facilidad. Después comprueba si la impresora está disponible en ese momento. Si no lo está, espera hasta que lo esté. Tan pronto como la impresora está disponible, el sistema operativo copia el primer carácter al registro de datos de la impresora.

```
copiar_del_usuario(bufer, p, cuenta);                  /* p es el búfer del kernel */
for (i=0; i<cuenta; i++) {                             /* itera en cada carácter */
        while (*reg_estado_impresora != READY);        /* itera hasta que esté lista */
        *registro_datos_impresora = p[i];              /* imprime un carácter */
}
regresar_al_usuario();
```

Fig. 2. Cómo escribir una cadena en la impresora usando E/S programada.

La E/S programada es simple, pero tiene la desventaja de ocupar la CPU tiempo completo hasta que se completen todas las operaciones de E/S.

C. E/S controlada por interrupciones

La forma de permitir que la CPU haga algo más mientras espera a que la impresora esté lista es utilizar interrupciones. Cuando se realiza la llamada al sistema para imprimir la cadena, el búfer se copia en espacio de kernel y el primero carácter se copia a la impresora, tan pronto como esté dispuesta para aceptar un carácter. La CPU llama al planificador y se ejecuta algún otro proceso. el proceso que pidió imprimir la cadena se bloquea hasta que se haya impreso toda la cadena. El trabajo realizado en la llamada al sistema se muestra en la Figura **??**.

```
copiar_del_usuario(bufer, p, cuenta);
habilitar_interrupciones();
while (*reg_estado_impresora != READY);
*registro_datos_impresora = p[0];
planificador();
```

```
if (cuenta==0) {
  desbloquear_usuario();
} else {
  *registro_datos_impresora = p[i];
  cuenta=cuenta – 1;
  i = i + 1;
}
reconocer_interrupcion();
regresar_de_interrupcion();
```

Fig. 3. Cómo escribir una cadena a la impresora, usando E/S controlada por interrupciones. (a) Código que se ejecuta al momento en que se hace una llamada al sistema para imprimir. (b) Procedimiento de servicio de interrupciones para la impresora.

D. E/S mediante el uso de DMA

Las interrupciones requieren tiempo, por lo que este esquema desperdicia cierta cantidad de tiempo de la CPU. Una solución es utilizar DMA. Aquí la idea es permitir que el controlador de DMA alimente los caracteres a la impresora uno a la vez, sin que la CPU se moleste. El DMA es E/S programada, sólo que el controlador de DMA realiza todo el trabajo en vez de la CPU principal. Esta estrategia requiere hardware especial pero libera la CPU durante la E/S para realizar otro trabajo.

La gran ganancia con DMA es reducir el número de interrupciones, de una por cada carácter a una por cada búfer impreso. Si hay muchos caracteres y las interrupciones son lentas, esto puede ser una gran mejora. El controlador de DMA es comúnmente más lento que la CPU principal. Si el controlador de DMA no puede controlar el dispositivo a toda su velocidad, o si la CPU por lo general no tiene nada que hacer mientras espera la interrupción de DMA, entonces puede ser mejor utilizar la E/S controlada por interrupción o incluso la E/S programada. De todas formas, la mayor parte del tiempo vale la pena usar DMA.

3. CAPAS DEL SOFTWARE DE E/S

EL software de E/S se organiza en cuatro capas. Cada capa tiene una función ben definida de realizar, y una interfaz bien definida

para los niveles adyacentes. La funcionalidad y las interfaces difieren de un sistema a otro.

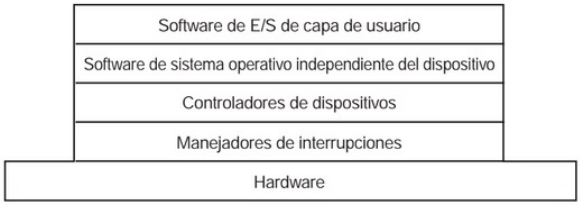

Fig. 4. Capas del sistema de software de E/S

A. Manejadores de interrupciones

Para la mayor parte de las operaciones de E/S las interrupciones son un hecho incómodo de la vida y no se pueden evitar. Deben ocultarse en la profundidad de las entrañas del sistema operativo, de manera que éste sepa lo menos posible de ellas. La mejor manera de ocultarlas es hacer que el controlador que inicia una operación de E/S se bloquee hasta que se haya completado la E/S y ocurra la interrupción. El controlador se puede bloquear a sí mismo realizando una llamada a down en un semáforo, una llamada a wait en una variable de condición, una llamada a receive en un mensaje o algo similar, por ejemplo.

Después puede desbloquear el controlador que la inició. en algunos casos sólo completará up en un semáforo. El efecto neto de la interrupción será que un controlador que estaba bloqueado podrá ejecutarse ahora. Este modelo funciona mejor si los controladores están estructurados como los procesos del kernel, con sus propios estados, pilas y contadores del programa.

B. Drivers de dispositivos

El número de registros de dispositivos y la naturaleza de los comandos varían radicalmente de un dispositivo a otro.

Cada dispositivo de E/S conectado a una computadora necesita cierto código específico para controlarlo. Este código, conocido como **driver**, es escrito por el fabricante del dispositivo y se incluye junto con el mismo. Como cada sistema operativo necesita sus propios drivers, los fabricantes de dispositivos por lo común proporcionan para varios sistemas operativos populares.

Cada driver maneja un tipo de dispositivo o, a lo más, una clase de dispositivos estrechamente relacionados.

Para poder utilizar el hardware del dispositivo, el driver por lo general tiene que formar parte del kernel del sistema operativo, cuando menos en las arquitecturas actuales. Este diseño aísla al kernel de los controladores, y a un controlador de otro, eliminando una fuente importante de fallas en el sistema. Para construir sistemas altamente confiables, ésta es, en definitiva, la mejor manera de hacerlo.

Las categorías más comunes son los **dispositivos de bloque** como los discos, que contienen varios bloques de datos que se pueden direccionar de manera independiente, y los **dispositivos de carácter** como los teclados y las impresoras, que generan o aceptan un flujo de caracteres.

La mayoría de los sistemas operativos definen una interfaz estándar que todos los controladores de bloque deben aceptar. Estas interfaces consisten en varios procedimientos que el resto del sistema operativo puede llamar para hacer que el controlador realice un trabajo para él. Los procedimientos ordinarios son los que se utilizan para leer un bloque o escribir una cadena de caracteres.

C. Software de E/S independiente del dispositivo

El límite exacto entre los controladores y el software independiente del dispositivo depende del sistema, debido a que ciertas funciones que podrían realizarse de una manera independiente al dispositivo pueden realizarse en los controladores, por eficiencia u otras razones.

Interfaz uniforme para controladores de dispositivos
Uso de búfer
Reporte de errores
Asignar y liberar dispositivos dedicados
Proporcionar un tamaño de bloque independiente del dispositivo

Fig. 5. Funciones del software de E/S independiente del dispositivo.

La función básica del software independiente del dispositivo es realizar las funciones de E/S que son comunes para todos los dispositivos y proveer una interfaz uniforme para el software a nivel de usuario. A continuación analizaremos las cuestiones antes mencionadas con más detalle.

Interfaz uniforme para los controladores de software de dispositivos

No es conveniente tener que modificar el sistema operativo para cada nuevo sistema operativo.

Otro aspecto de tener una interfaz uniforme es la forma en que se nombran los dispositivos. el software independiente del dispositivo se hace cargo de asignar nombres de dispositivo simbólicos al controlador apropiado. En UNIX el nombre de un dispositivo como /dev/disk0 especifica de manera única el nodo-i para un archivo especial, y este nodo-i contiene el **número mayor de dispositivo**, que se utiliza para localizar el controlador apropiado. El nodo-i también contiene **número menor de dispositivo**, que se pasa como un parámetro al controlador para poder especificar la unidad que se va a leer o escribir. Todos los dispositivos tienen números mayores y menores, y para acceder a todos los controladores se utiliza el número mayor de dispositivo para seleccionar el controlador.

Uso de búfer

El uso de búfer es otra cuestión, tanto para los dispositivos de bloque como los de carácter, por una variedad de razones.

Reporte de errores

Los errores son mucho más comunes en el contexto de la E/S que en otros. Muchos errores son específicos de cada dispositivo y el controlador apropiado debe manejarlos, pero el marco de trabajo para el manejo de errores es independiente del dispositivo.

Los errores de programación son una clase de errores de E/S. Éstos ocurren cuando un proceso pide algo imposible, como escribir en un dispositivo de entrada o leer de un dispositivo de salida. Otros errores son proporcionar una dirección de búfer inválida o algún otro parámetro, y especificar un dispositivo inválido, entre otros. La acción a tomar en estos errores es simple: sólo se reporta un código de error al que hizo la llamada.

D. Sosftware de E/S en espacio de usuario

Aunque la mayor parte del software de E/S está dentro del sistema operativo, una pequeña porción de éste consiste en bibliotecas vinculadas entre sí con programas de usuario, e incluso programas enteros que se ejecutan desde el exterior del kernel. Las llamadas al sistema, incluyendo las llamadas al sistema de E/S, se realizan comúnmente mediante procedimientos de biblioteca.

No todo el software de E/S de bajo nivel consiste en procedimientos de biblioteca. Otra categoría importante es el sistema de colas. El **uso de colas** es una manera de lidiar con los dispositivos de E/S dedicados en un sistema de multiprogramación. Considere un dispositivo común que utiliza colas: una impresora. Suponga que un proceso lo abriera y no hiciera nada durante horas. Ningún proceso podría imprimir nada.

Lo que se hace es crear un proceso especial, conocido como **demonio**, y un directorio especial llamado **directorio de cola de impresión**. Al proteger el archivo especial contra el uso directo por parte de los usuarios, se elimina el problema de que alguien lo mantenga abierto por un tiempo innecesariamente extenso.

También se utiliza en otras situaciones de E/S. Por ejemplo, la transferencia de archivos a través de la red utiliza con frecuencia un demonio de red.

Empezando desde la parte inferior, los niveles son el hardware, los manejadores de interrupciones, los controladores de dispositivos, el software independiente del dispositivo y, por último, los procesos del usuario.

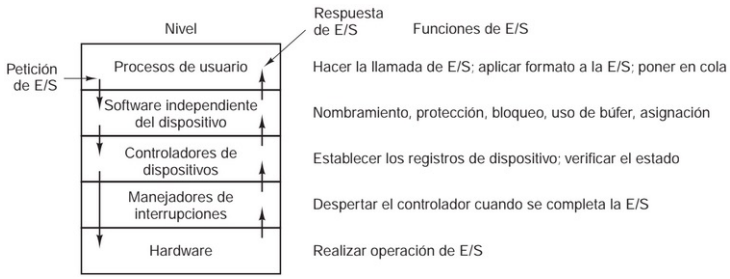

Fig. 6. Niveles del sistema de E/S y las funciones principales de cada nivel.

Cuando un programa de usuario trata de leer un bloque de un archivo, se invoca el sistema operativo para llevar a cabo una llamada. el software independiente del dispositivo busca el bloque en la caché del búfer, por ejemplo. Si el bloque necesario no está ahí, llama al controlador del dispositivo para enviar la petición al hardware y obtenerlo del disco. Después el proceso se bloquea hasta que se haya completado la operación del disco.

Cuando termina el disco, el hardware genera una interrupción. El manejador de interrupciones se ejecuta para descubrir qué ocurrió; es decir, qué dispositivo desea ateción en ese momento. Después extrae el estado del dispositivo y despierta al proceso inactivo para que termine la petición de E/S y deje que el proceso de usuario continúe.

REFERENCES

1. Andrew S. Tanenbaum, Sistemas Operativos Modernos, Tercera Edición, Ed. Pearson (2009).
2. Silberschatz Galvin, Sistemas Operativos, Quinta Edición, Ed. Addison Wesley (1999).

SOBRE MI

Joaquín Roiz Pagador

Estudiante del Grado en Ingeniería Informática en Sistemas de Información en la universidad *Pablo de Olavide*, promoción de 2016.