Sistemas Operativos

Subsistema de Entrada/Salida

Curso 2024

Facultad de Ingeniería, UDELAR

Agenda

1. Introducción

2. Métodos para realizar una E/S

3. Interfaz de aplicación de E/S

4. Subsistema de E/S

Introducción

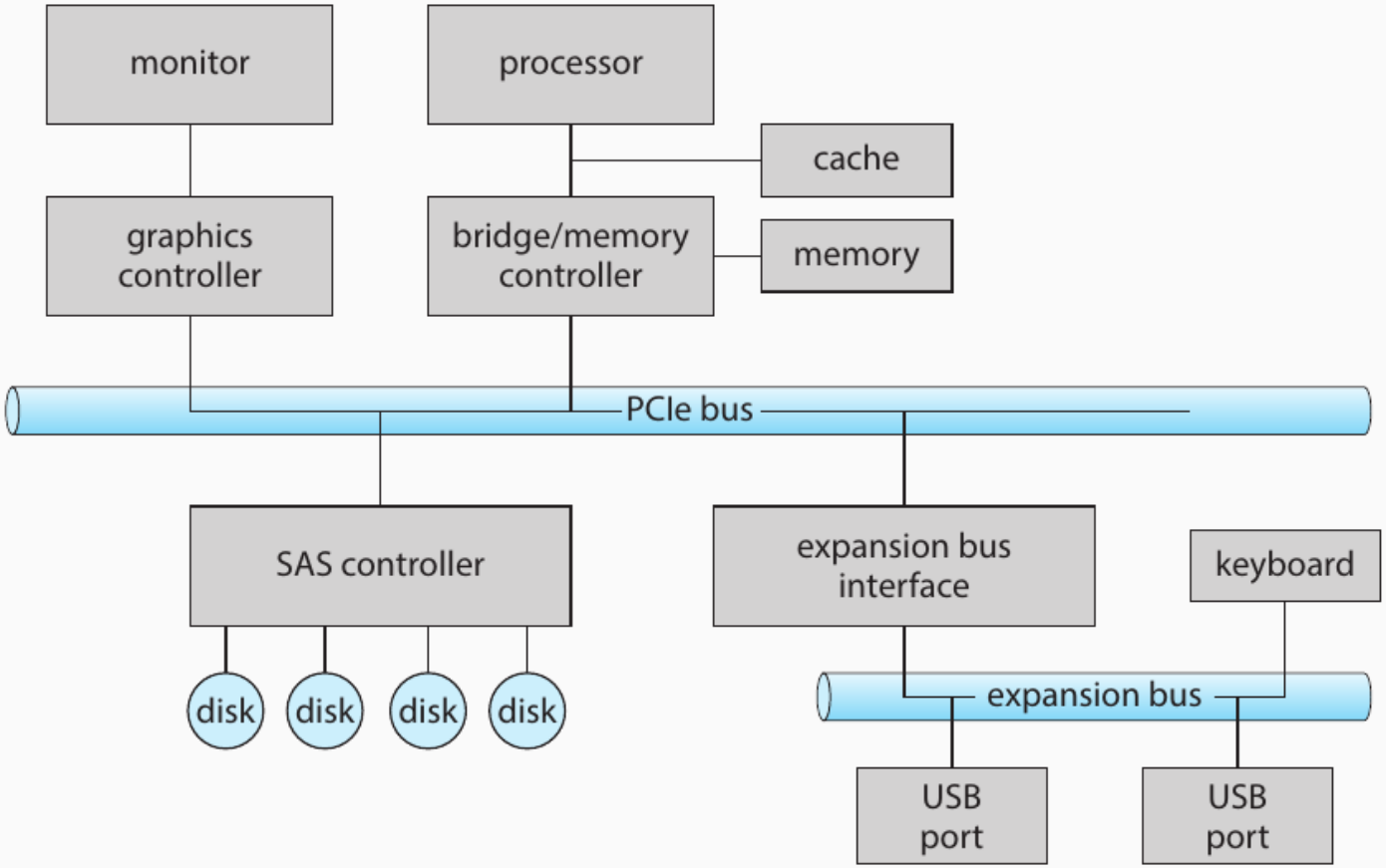
Introducción

- Uno de los principales funciones de un sistema operativo es controlar todos los dispositivos de Entrada/Salida que estén conectados al mismo: teclado, ratón, impresora, monitor, red, disco, etc.
- Para encapsular los detalles de los diferentes dispositivos, el núcleo del sistema operativo es estructurado en el uso de módulos de dispositivos.
- Los manejadores de dispositivos (*device drivers*) presentan un acceso uniforme al subsistema de Entrada/Salida.

Introducción

- Los dispositivos se comunican con la computadora a través de señales sobre un puerto.
- Si varios dispositivos utilizan el mismo medio de comunicación, la conexión es llamada bus.
- Un bus es un conjunto de líneas (*wires*) y un protocolo que especifica un conjunto de mensajes que son enviados a través de él.
- Los buses son comúnmente usados en los sistemas de computación para interconectar los dispositivos

Introducción



Introducción

- Un controlador (o adaptador) es un chip electrónico que puede operar en un puerto, bus, o dispositivo.
- Por ejemplo, un controlador de puerto serial es un dispositivo que controla las señales sobre un puerto serial.
- Muchos dispositivos tienen implementado su propio controlador que está incorporado al dispositivo (ej.: discos).
- La comunicación con el controlador por parte del procesador es a través de registros de datos y señales de control. El procesador se comunica con el controlador escribiendo y leyendo información en los registros del controlador.

Introducción

- Una vía de comunicación es a través de instrucciones de E/S especiales que especifican una transferencias de un byte, o palabra hacia una dirección de puerto de E/S (*I/O port address*).
- La instrucción de E/S genera las señales adecuadas sobre las líneas del bus para lograr la comunicación con el dispositivo que se intenta acceder y mover bits hacia y fuera de los registros del dispositivo.
- Otra alternativa es utilizar mapeo de memoria del dispositivo de E/S (*memory-mapped I/O*). Los registros del dispositivos son “mapeados” a memoria principal.
    - Un ejemplo de este tipo de comunicación es el controlador de una tarjeta de video.

Introducción

- Un puerto de E/S consiste, generalmente, en cuatro registros:
  - Estado (*status*): estos registros pueden ser leídos por el equipo. Informan del estado del dispositivo (completitud de una operación, disponibilidad de leer del registro de datos de entrada, existencia de un error en el dispositivo)
  - Control (*control*): pueden ser escritos por el equipo para generar un pedido, cambiar de modo el dispositivo.
  - Datos de entrada (*data-in*): son leídos por el equipo para obtener la entrada.
  - Datos de salida (*data-out*): son escritos por el equipo para enviar una salida.

Métodos para realizar una E/S

Métodos para realizar una E/S

- E/S Programada (*Programmed I/O*): El procesador le comunica un pedido a la controladora del dispositivo y queda en un busy waiting consultando a la controladora para verificar el estado del pedido.
- Interupcciones (*Interrupt-Driven I/O*): El procesador le comunica el pedido a la controladora y se libera para realizar otras tareas. Al culminar el pedido el dispositivo, la controladora genera una interrupción al procesador.
- Acceso directo a memoria (*DMA – Direct Memory Access*): Se utiliza un chip especial que permite transferir datos desde alguna controladora a memoria sin que el procesador tenga que intervenir en forma continua.

E/S Programada (Programmed I/O)

- El procesador genera una solicitud de E/S y luego se encarga de controlar la completitud de la misma controlando algún registro del controlador de dispositivo.
- La consulta la realiza en una iteración continua denominada polling o busy waiting.
- La técnica de busy waiting consume ciclos de procesador en forma innecesaria.

E/S Programada (Programmed I/O)

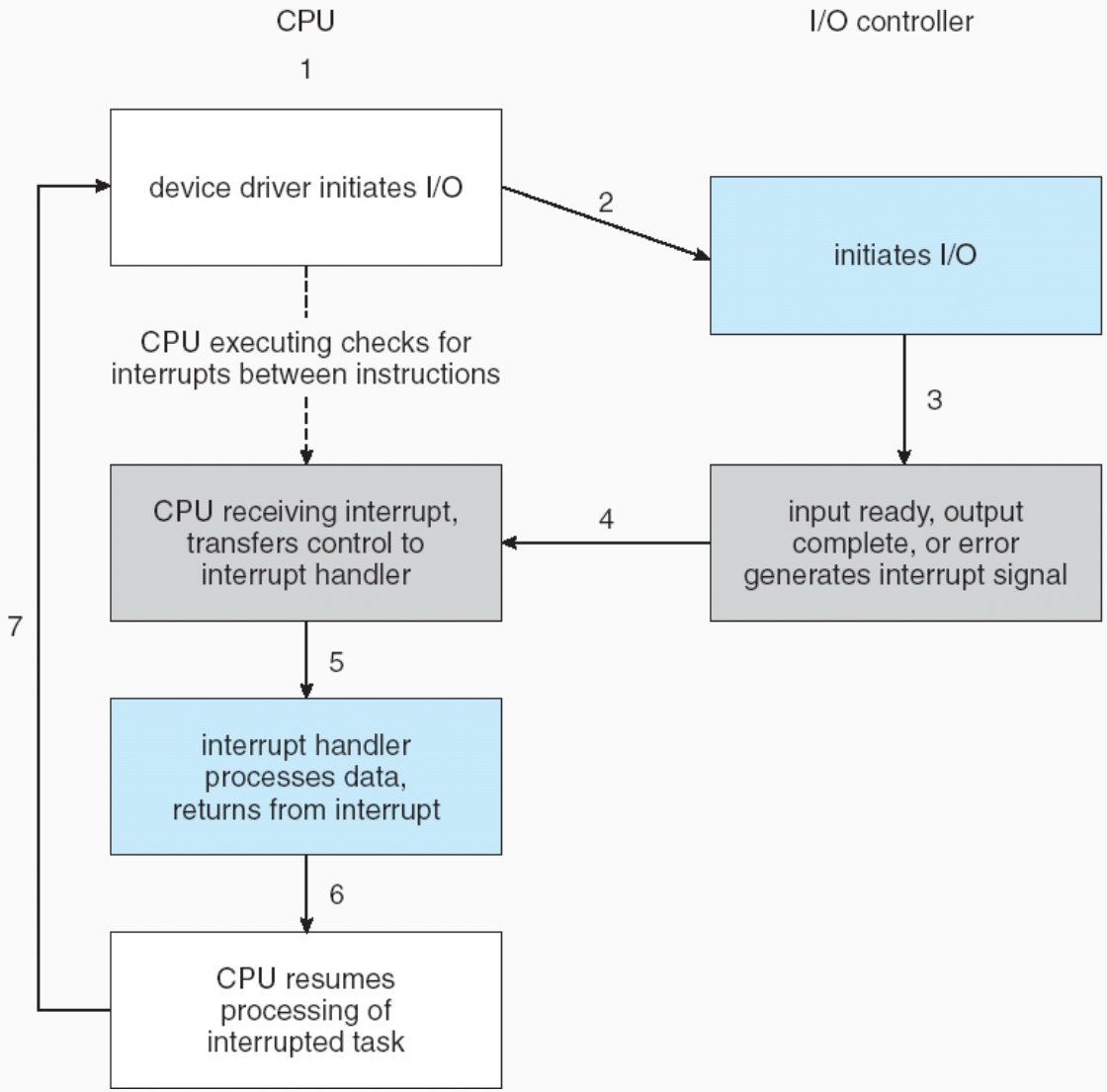
```
p = copy_from_user(buffer, k_buffer, count);
for (i = 0; i < count; i++) {
        while (*printer_status_reg != READY);
        *printer_data.register = p[i];
}
return_to_user();
```

- Si bien permite una programación simple, tiene como gran desventaja el desperdicio de ciclos de procesador.

Interrupciones (Interrupt-Driven I/O)

- El proceso que está ejecutando realiza la solicitud de E/S, se agrega a la cola de espera del dispositivo y, finalmente, invoca al planificador (*scheduler*) para que asigne el procesador a otro proceso.
- El controlador de dispositivo avisará la completitud de la solicitud a través de una interrupción. Una rutina de atención de la interrupción será invocada interrumpiendo la ejecución del proceso asignado al procesador. Es necesario salvar el estado del proceso que estaba ejecutando.
- Una vez que la completitud de la E/S es registrada por el manejador de la interrupción (*interrupt handler*), el proceso que generó la solicitud es desbloqueado y se lo asigna a la lista de procesos listos.

Interrupciones (Interrupt-Driven I/O)



Interrupciones (Interrupt-Driven I/O)

```
p = copy_from_user(buffer, k_buffer, count);
habilitar_interrupciones();
while (*printer_status_reg != READY);
*printer_data.register = p[0];
add_to_queue(current);
scheduler();
remove_from_queue(current);
```

- La rutina scheduler asignará el procesador a otro proceso. Cuando este proceso vuelva a ejecutar, lo hará luego de la instrucción scheduler.

Interrupciones (Interrupt-Driven I/O)

```
if (count == 0)
    unblock_user();
else {
    *printer_data_register = p[i];
    count--;
    i++;
}
acknowledge_interrupt();
return_from interrupt();
```

- La gran ventaja es que el procesador queda disponible para otro proceso que requiera el recurso. Es necesario soporte de hardware, además, se deben codificar los manejadores de interrupciones (*interrupt handlers*)

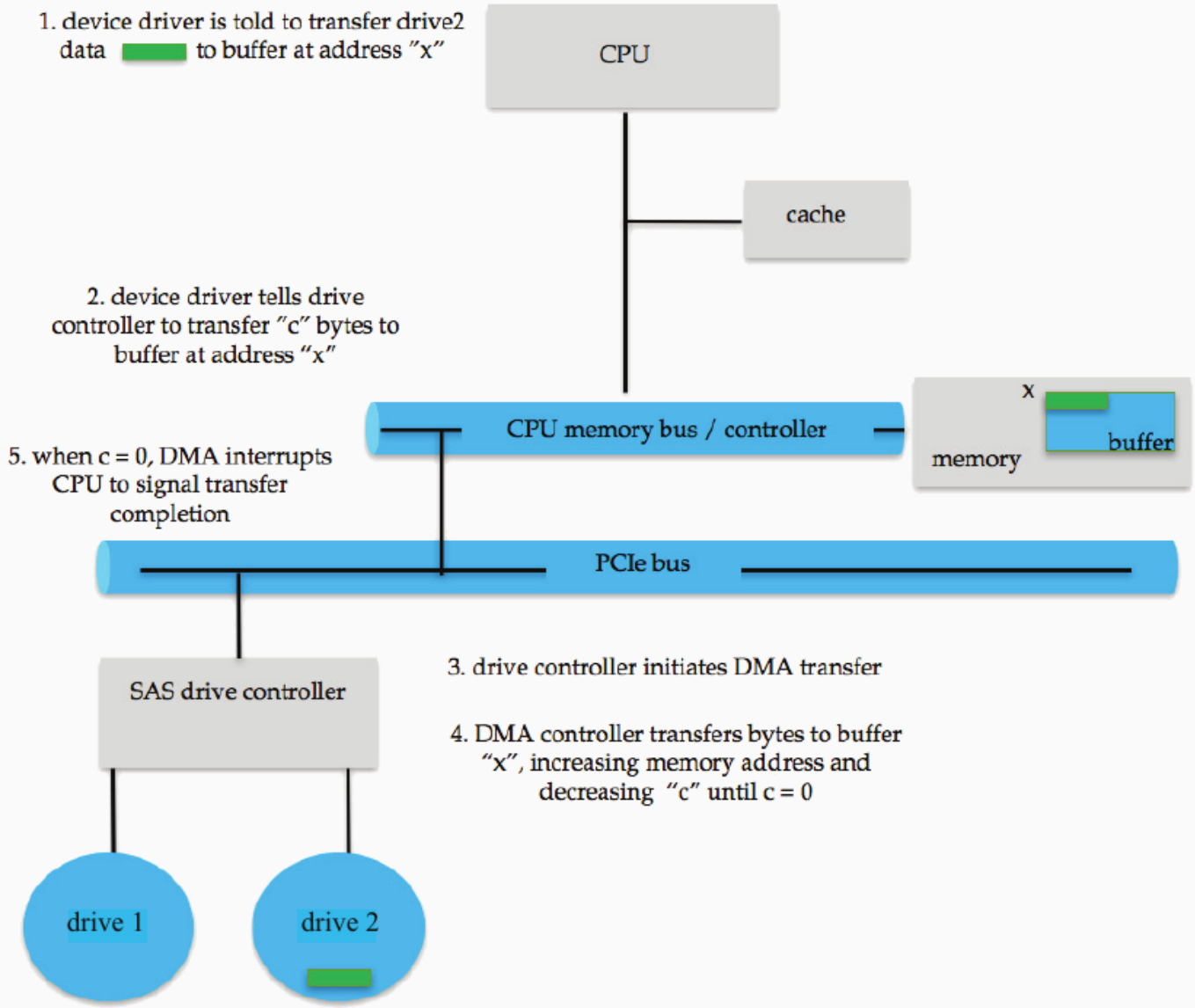
Interrupciones (Interrupt-Driven I/O)

vector number	description
0	divide error
1	debug exception
2	null interrupt
3	breakpoint
4	INTO-detected overflow
5	bound range exception
6	invalid opcode
7	device not available
8	double fault
9	coprocessor segment overrun (reserved)
10	invalid task state segment
11	segment not present
12	stack fault
13	general protection
14	page fault
15	(Intel reserved, do not use)
16	floating-point error
17	alignment check
18	machine check
19–31	(Intel reserved, do not use)
32–255	maskable interrupts

Acceso Directo a Memoria (Direct Memory Access) DMA

- Un problema que presenta en el manejo por interrupciones es que, por ejemplo, los dispositivos orientados a caracteres interrumpen en cada transferencia. Esto puede generar una degradación importante del sistema.
- Se dispone de un dispositivo especializado que permite realizar transferencias desde ciertos dispositivos a memoria. La transferencia se hace en paralelo mientras el procesador realiza otras tareas.
- El procesador carga ciertos registros en el controlador DMA para realizar el pedido. El controlador DMA se encarga de la tarea de transferencia, interrumpiendo al procesador cuando finalizó.

Acceso Directo a Memoria (Direct Memory Access) DMA



Acceso Directo a Memoria (Direct Memory Access) DMA

```
p = copy_from_user(buffer, p, count);
set_up_DMA_controller();
add_queue_dma(current);
scheduler();
remove_queue_dma(current);
...
```

Rutina de interrupción:

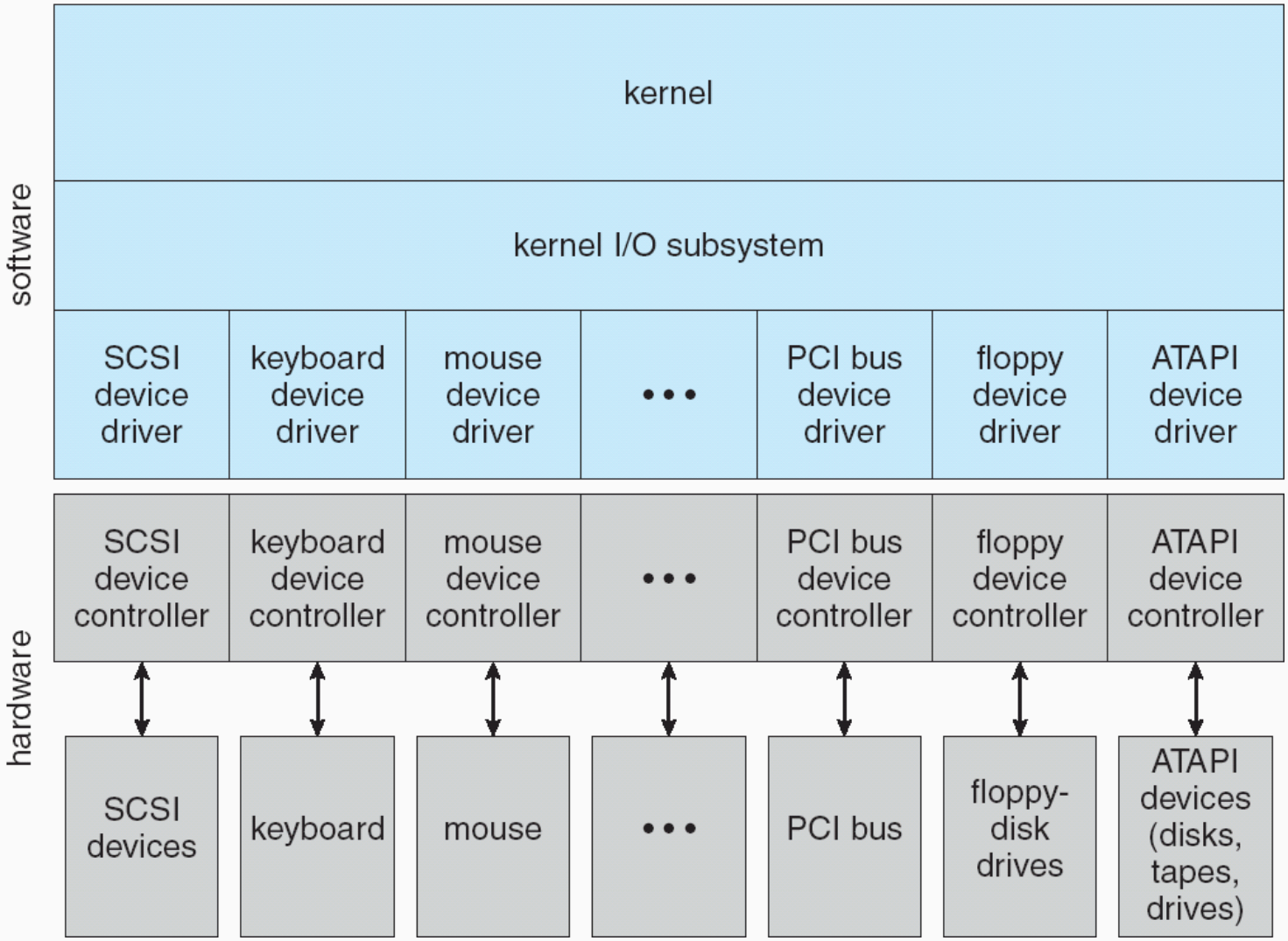
```
unblock_user();
return_from_interrupt();
```

Interfaz de aplicación de E/S

Interfaz de aplicación de E/S

- El sistema operativo propone estructuras e interfaces para lograr manipular dispositivos de E/S de forma estándar y uniforme.
- Esta aproximación genera abstracción, encapsulamiento y utilización de capas en el software.
- Se abstraerá los dispositivos identificando sus características y proponiendo una forma uniforme de acceso a los dispositivos del mismo tipo.
- Cada dispositivo (si bien pueden pertenecer a un mismo tipo) encapsulará la comunicación con la controladora en módulos llamados device drivers independientes.
- Lograr independizar el subsistema de E/S del hardware simplifica el trabajo a los desarrolladores de un sistema operativo.

Interfaz de aplicación de E/S



Interfaz de aplicación de E/S

- Características de los dispositivos de E/S

aspect	variation	example
data-transfer mode	character block	terminal disk
access method	sequential random	modem CD-ROM
transfer schedule	synchronous asynchronous	tape keyboard
sharing	dedicated sharable	tape keyboard
device speed	latency seek time transfer rate delay between operations	
I/O direction	read only write only read–write	CD-ROM graphics controller disk

Interfaz de aplicación de E/S

- Dispositivos de bloques (*Block devices*)
  - Son dispositivos que su granularidad de información es a nivel de bloques. Se especializan en transferir grandes volúmenes de datos.
  - Ej.: Discos.
- Dispositivos de caracteres (*Char devices*)
  - La granularidad es a través de caracteres.
  - Ej.: teclados, ratón, puertos serial.

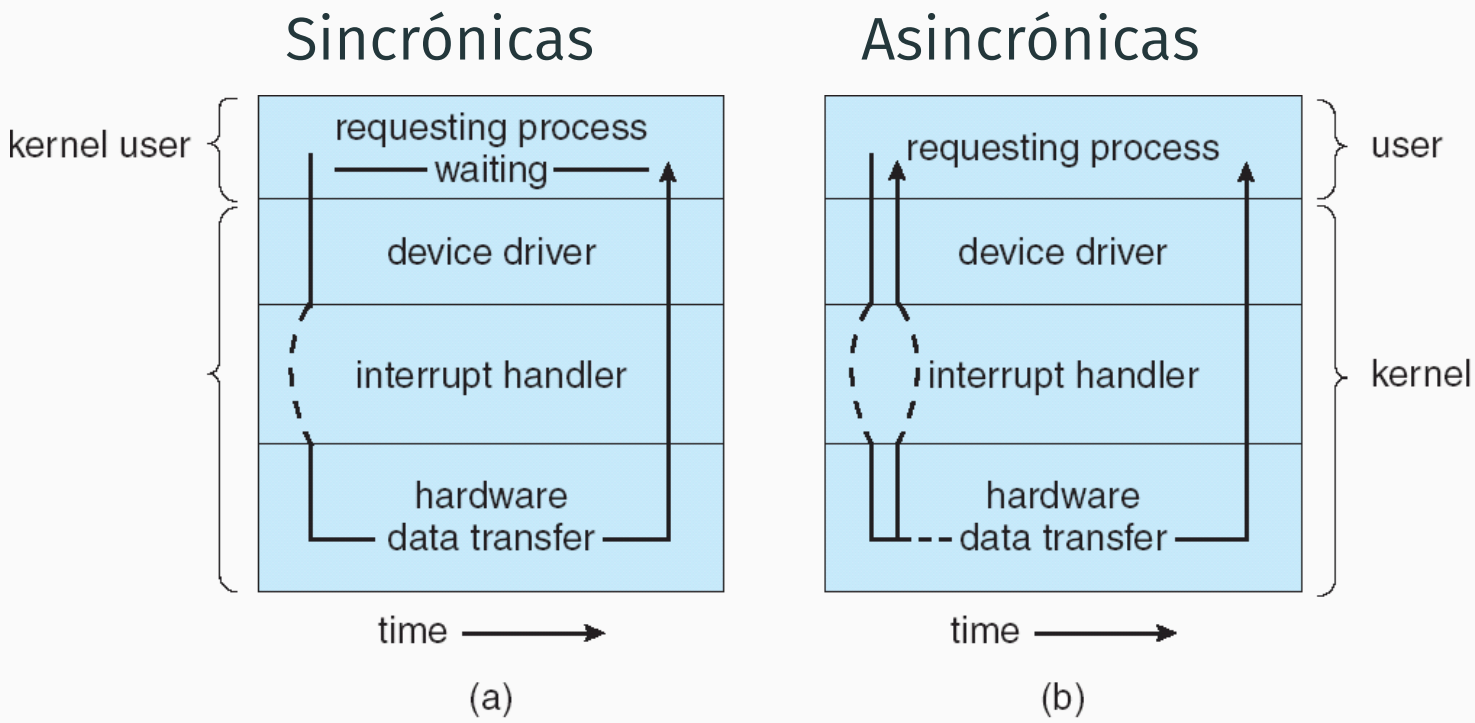
Interfaz de aplicación de E/S

- Operaciones bloqueantes (*Blocking*)
  - Cuando un proceso requiere de un servicio de E/S a través de una rutina bloqueante, el proceso se suspende hasta que la operación haya finalizado.
  - Es fácil de utilizar y entender.
- Operaciones no bloqueantes (*Nonblocking*)
  - El llamado al servicio de E/S es devuelto tan pronto como sea posible.
  - Ej. un read sobre un archivo traerá quizás algunos datos, pero no necesariamente todos.

Interfaz de aplicación de E/S

- Operaciones asincrónicas (*asynchronous*)
  - La operación es ejecutada en paralelo. Cuando finaliza le avisa a través de una señal. Son difíciles de utilizar.
- Operaciones sincrónicas (*synchronous*)
  - Los procesos que realizan pedidos de E/S se bloquean hasta que el pedido finalice.

Interfaz de aplicación de E/S



Subsistema de E/S

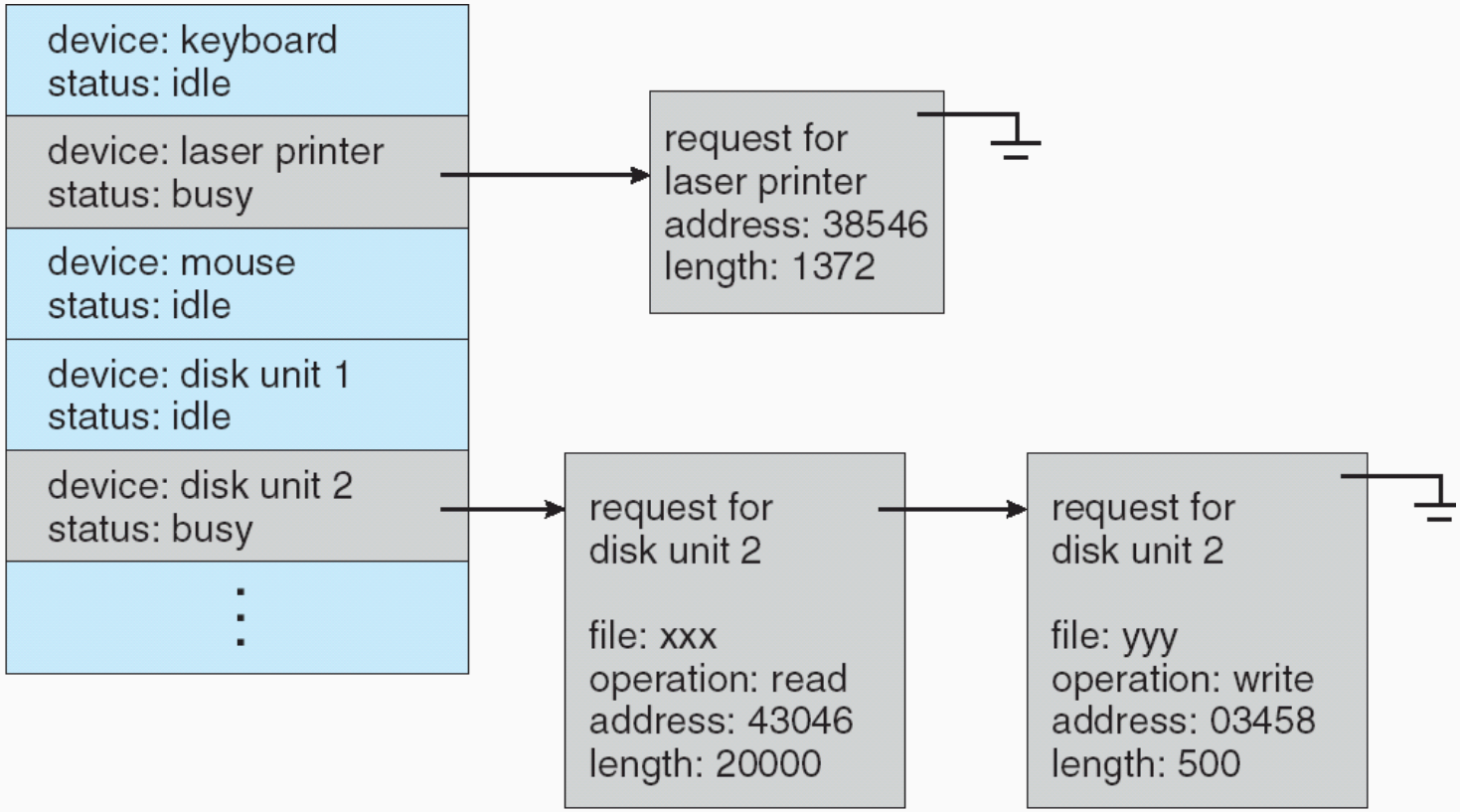
Subsistema de E/S

- El núcleo del sistema operativo brinda varios servicios para el manejo de E/S, que están desarrollados en la infraestructura de hardware y device drivers:
  - Planificación de E/S (*I/O Scheduling*).
  - Buffering.
  - Caching.
  - Spooling.
  - Manejo de errores (*Error handling*).

Planificación de E/S (I/O Scheduling)

- La planificación de requerimientos de E/S pretende lograr un buen rendimiento del dispositivo.
- Seguramente los pedidos generados a través de llamados a sistema por parte de los procesos, no generan una buena secuencia de planificación para el dispositivo.
- El sistema operativo implementa la planificación manteniendo una cola de pedidos por cada dispositivo.
- Cuando un proceso genera una E/S bloqueante, es puesto en la cola del dispositivo correspondiente y el subsistema de E/S reorganiza los pedidos para lograr un mayor rendimiento.

Planificación de E/S (I/O Scheduling)



Buffering

- Es un lugar de memoria que guarda información (datos) mientras son transferidos entre dos dispositivos o un dispositivo y una aplicación.
- Existen 3 razones para realizar buffering:
  - Normalizar las velocidades entre diferentes dispositivos.
  - Adaptarse entre dispositivos que difieren en los tamaños de transferencia.
  - Mantener la semántica de aplicaciones que realizan E/S: en una operación write el buffer de usuario es copiado a un buffer del sistema operativo. De esa forma, el sistema logra independizarse de la aplicación.

Caching

- La cache es una región de memoria más rápida que contiene copias de datos.
- Su utilidad es acelerar el acceso a la información.
- En buffering se tiene los datos originales, mientras que caching se tiene una o varias copias en un medio de memoria más rápido.
- El caching introduce problemas de consistencia de la información.

Spooling

- Es un buffer que mantiene salida para un dispositivo que no se pueda intercalar.
- El sistema captura la salida para el dispositivo y la va guardando para brindarla en forma correcta (sin intercalar).
- El spooling es una forma que el sistema operativo tiene para coordinar salida concurrente para dispositivos.
- Tiene la ventaja que libera al proceso, permitiendo continuar su ejecución. Su trabajo es guardado y será enviado al dispositivo cuando el subsistema lo crea conveniente.
- Ej.: impresora.

Manejo de errores (Error handling)

- Cada llamado a sistema retorna un bit que informa el éxito o fracaso de la operación sobre una E/S.
- En UNIX se utiliza una variable errno que es utilizada para codificar el error.
- Algunos dispositivos permiten obtener información detallada de la naturaleza de un error (ej: controladora SCSI)