

Tema 5:

Gestión de entradas-salidas



José Antonio Gómez Hernández, 2020.

Contenidos

- ▷ Arquitectura software del sistema de entradas/salidas.
- ▷ Archivos de dispositivos.
- ▷ Manejadores de dispositivos.





0.

Hardware de E/S

Repaso de conceptos previos

Hardware de E/S

- ▷ Elementos hardware del sistema de E/S y arquitectura:
 - Bus
 - Puerto
 - Controlador
 - Dispositivo
- ▷ Comunicación con el hardware de E/S:
 - Sondeo
 - Interrupciones
 - DMA
 - Dispositivos proyectados en memoria



Arquitectura hardware

- ▷ El hardware asociado con un dispositivo de E/S consta de cuatro elementos básicos:
 - Un **bus** para comunicarse con la CPU y es compartido entre varios dispositivos.
 - Un **puerto** que consta de varios registros:
 - **Estado** - indica si esta ocupado, los datos están listos, o ha ocurrido un error.
 - **Control** - operación que ha de realizar.
 - **Datos_entrada** - datos a enviar a CPU.
 - **Datos_salida** - datos recibidos de la CPU.

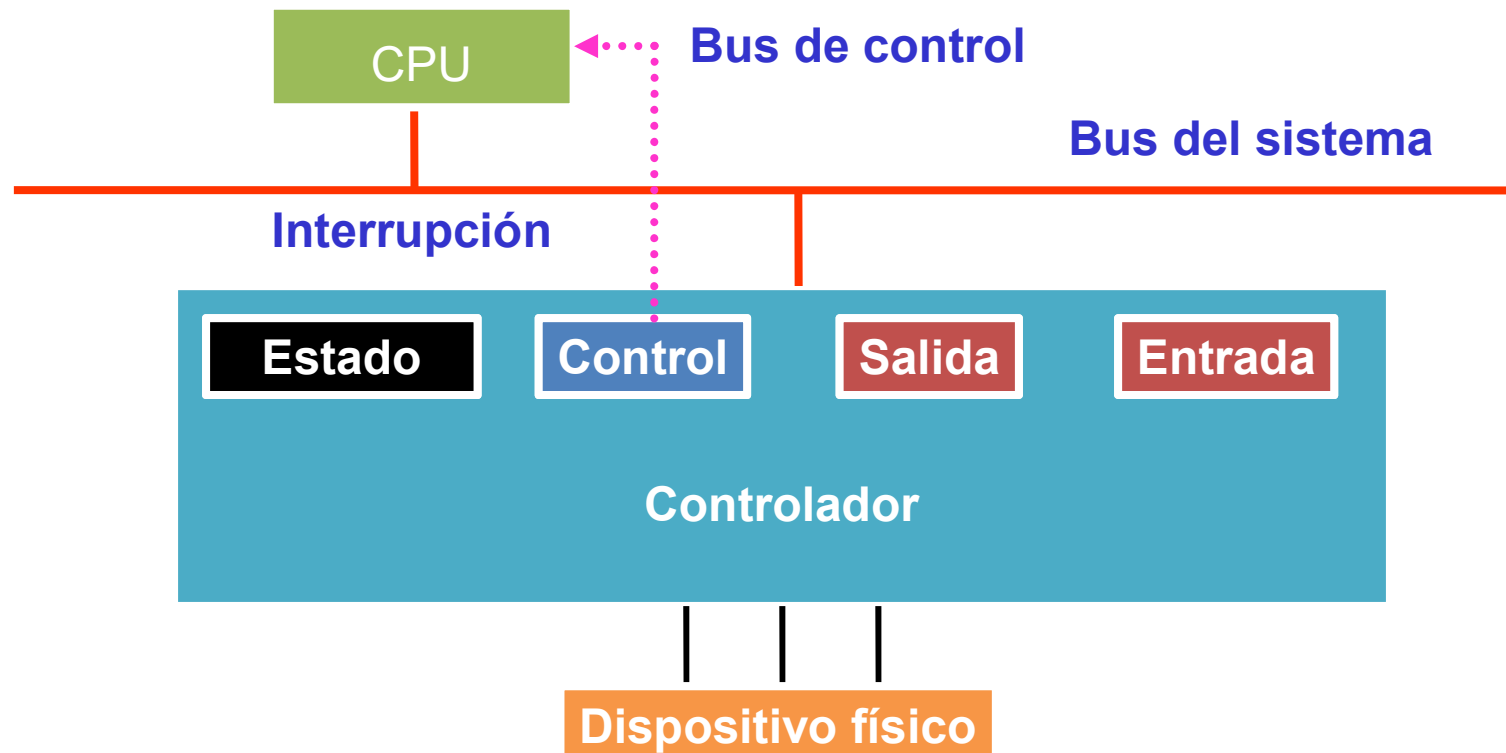


Arquitectura hardware (ii)

- Un **controlador** que recibe ordenes del bus del sistema, traduce ordenes en acciones del dispositivo, y lee/escribe datos desde/en el bus del sistema.
 - El propio dispositivo.
- ▷ Existen una gran variedad de dispositivos muy diferentes entre sí:
- Tradicionales: discos, impresoras, teclado, modem, ratón, pantalla, etc.
 - No tradicionales: joystick, actuador de robot, superficie de vuelo de un avión, sistema de inyección de un coche, etc.



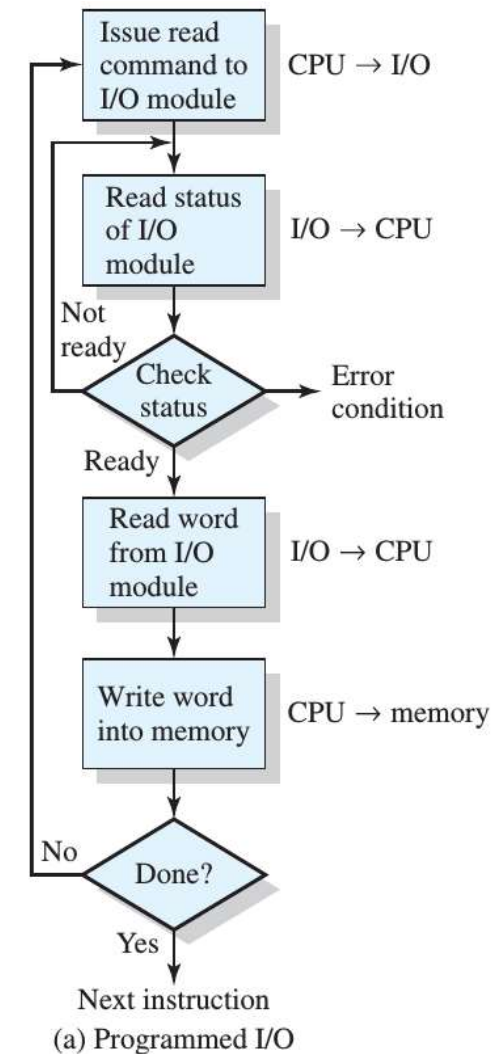
Arquitectura hardware: esquema



Comunicación dispositivo-SO: sondeo

▷ Los pasos a seguir en el sondeo:

1. CPU espera hasta que el estado sea *libre*.
2. CPU ajusta el registro de *ordenes y datos* de entrada o salida.
3. CPU ajusta el estado a *orden-preparada*.
4. El controlador reacciona a *orden-preparada* y pone estado a *ocupado*. Lee registro de *ordenes* y ejecuta orden, pone un valor en registros de *datos* si es necesario. Pone estado a *listo*.
5. Suponiendo que la orden tiene éxito, el controlador cambia el estado a ocioso.
6. La CPU observa el cambio a ocioso y lee los datos si es una operación de salida.



Stallings2009, Fig. 1.19a

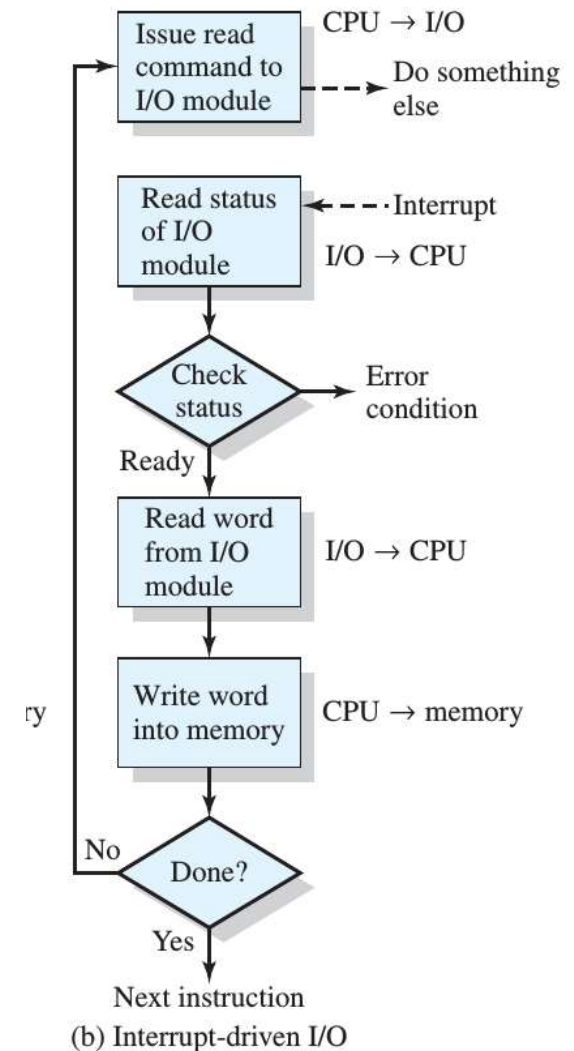
Sondeo: pros y contras

- ▷ Esta técnica es una buena elección si los datos van a ser manejados al instante (un modem o teclado), ya que los datos se perderían si no se retiran del dispositivo lo suficientemente rápido; pero ¿qué ocurre si el dispositivo es lento comparado con la CPU?



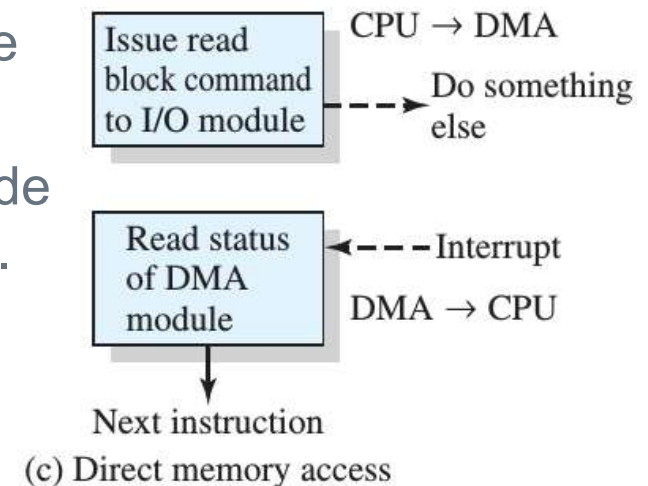
Comunicación dispositivo-SO: interrupción

- ▷ En lugar de tener la CPU ocupada esperando la disponibilidad del dispositivo, el dispositivo *interrumpe* a la CPU cuando ha terminado una operación de E/S.
- ▷ Cuando se produce la interrupción de E/S:
 1. Hay que determinar que dispositivo la provocó para poder asignar la RSI correspondiente.
 2. Se determina el estado de finalización. Si la última orden fue una operación de entrada, hay además que recuperar los datos del registro del dispositivo
 3. Inicia la siguiente operación para el dispositivo.



Comunicación dispositivo-SO: DMA (Direct Memory Access)

- ▷ Recuperar información byte a byte no adecuado para dispositivos que transfieren grandes cantidad de datos.
- ▷ **DMA** – Controlador de dispositivo que puede leer/escribir directamente en memoria. En lugar de registros de e/s, tiene un registro de dirección:
 1. La CPU indica al DMA la ubicación de la fuente/destino de la transferencia.
 2. DMA opera el bus e interrumpe a la CPU (robo de ciclo) cuando se completa la transferencia.
 3. DMA y CPU compiten por el bus de memoria.





1.

Arquitecturas software del sistema de E/S

Cómo se estructura el software destinado a gestionar las E/S

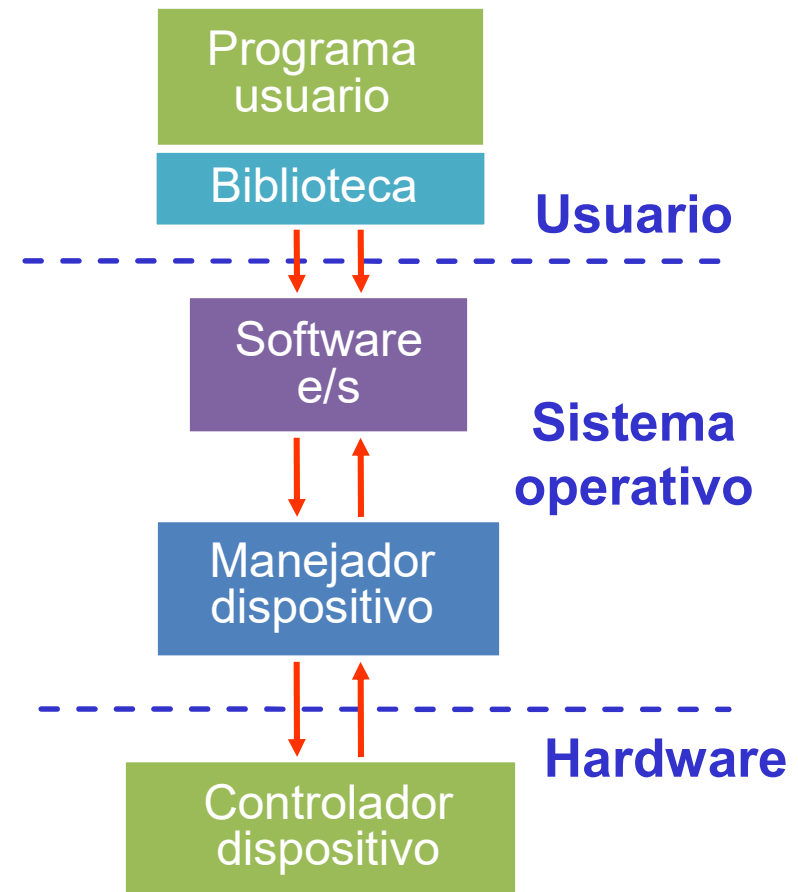
Servicios de E/S

- ▷ Designación de archivos y dispositivos.
- ▷ Control de acceso.
- ▷ Operaciones adecuadas para archivos y dispositivos.
- ▷ Asignación de dispositivos.
- ▷ *Búfering, caché, y spooling*, para suministrar una comunicación eficiente con el dispositivo.
- ▷ Planificación de E/S.
- ▷ Gestión de errores y recuperación de fallos asociados con el dispositivo.
- ▷ Aislar en un módulo las características y conducta específica del dispositivo.



Servicios de E/S

- ▷ Podemos estructurar el software de E/S en capas:
- Manejadores de dispositivos
 - Software de e/s independiente del dispositivo
 - Software a nivel de usuario.



Manejador de dispositivo

- ▷ Contienen todo el código dependiente del dispositivo. Cada manejador gestiona un tipo o clase de dispositivo.
- ▷ Acepta peticiones “abstractas” de la capa de software independiente del dispositivo, y controla que la petición se realiza:
 - Traduce petición abstracta en ordenes para el controlador del dispositivo.
 - Se bloquea o no, según tipo de operación.
 - Si no hay errores, da respuesta si es necesario, y retorna al llamador.





2.

Archivos de dispositivos

Cómo acceder a los dispositivos a través de la interfaz de archivos

Software independiente del dispositivo

- ▷ Realiza tareas comunes a todos los dispositivos y suministra una interfaz común al usuario.
- ▷ Las principales funciones son:
 - Suministra interfaz uniforme a los manejadores
 - Realiza la designación de dispositivos
 - Implementa la protección de dispositivos
 - Establece el tamaño de bloque independiente del dispositivo y asigna almacenamiento (para dispositivos de bloques)
 - Informar de errores producidos en las e/s.



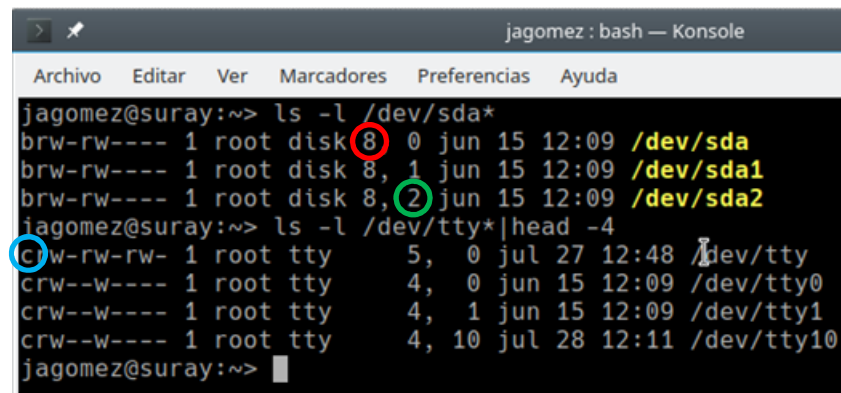
Designación de dispositivos

- ▷ Espacio de nombre de dispositivos - define cómo identificar y nombrar los dispositivos
- ▷ Existen diferentes espacios de nombre:
 - **Espacio de nombres hardware** - especifica el dispositivo por el controlador al que esta ligado y el número de dispositivo lógico dentro del controlador.
 - **Espacio de nombres kernel** - utilizado por el núcleo, suele basarse en el anterior.
 - **Espacio de nombres de usuario** - debe ser un esquema sencillo y familiar.



Designación (ii)

- ▷ El sistema de e/s independiente del dispositivo define las semánticas de los espacios de nombres kernel y usuario, y establece las correspondencias entre ellas.
- ▷ En Unix, el espacio de nombres kernel identifica un dispositivos por:
 - **Número principal** (*mayor*) - Identifica el controlador
 - **Número secundario** (*minor*) - Instancia del dispositivo.
 - **Tipo de dispositivo** – de *caracteres* o *bloques*.



The screenshot shows a terminal window titled 'jagomez : bash — Konsole'. The user has run the command 'ls -l /dev/sda*' and the output is as follows:

```
jagomez@suray:~> ls -l /dev/sda*
brw-rw---- 1 root disk 8, 0 jun 15 12:09 /dev/sda
brw-rw---- 1 root disk 8, 1 jun 15 12:09 /dev/sda1
brw-rw---- 1 root disk 8, 2 jun 15 12:09 /dev/sda2
```

Then, the user has run the command 'ls -l /dev/tty*|head -4' and the output is:

```
jagomez@suray:~> ls -l /dev/tty*|head -4
crw-rw-rw- 1 root tty 5, 0 jul 27 12:48 /dev/tty
crw--w---- 1 root tty 4, 0 jun 15 12:09 /dev/tty0
crw--w---- 1 root tty 4, 1 jun 15 12:09 /dev/tty1
crw--w---- 1 root tty 4, 10 jul 28 12:11 /dev/tty10
```

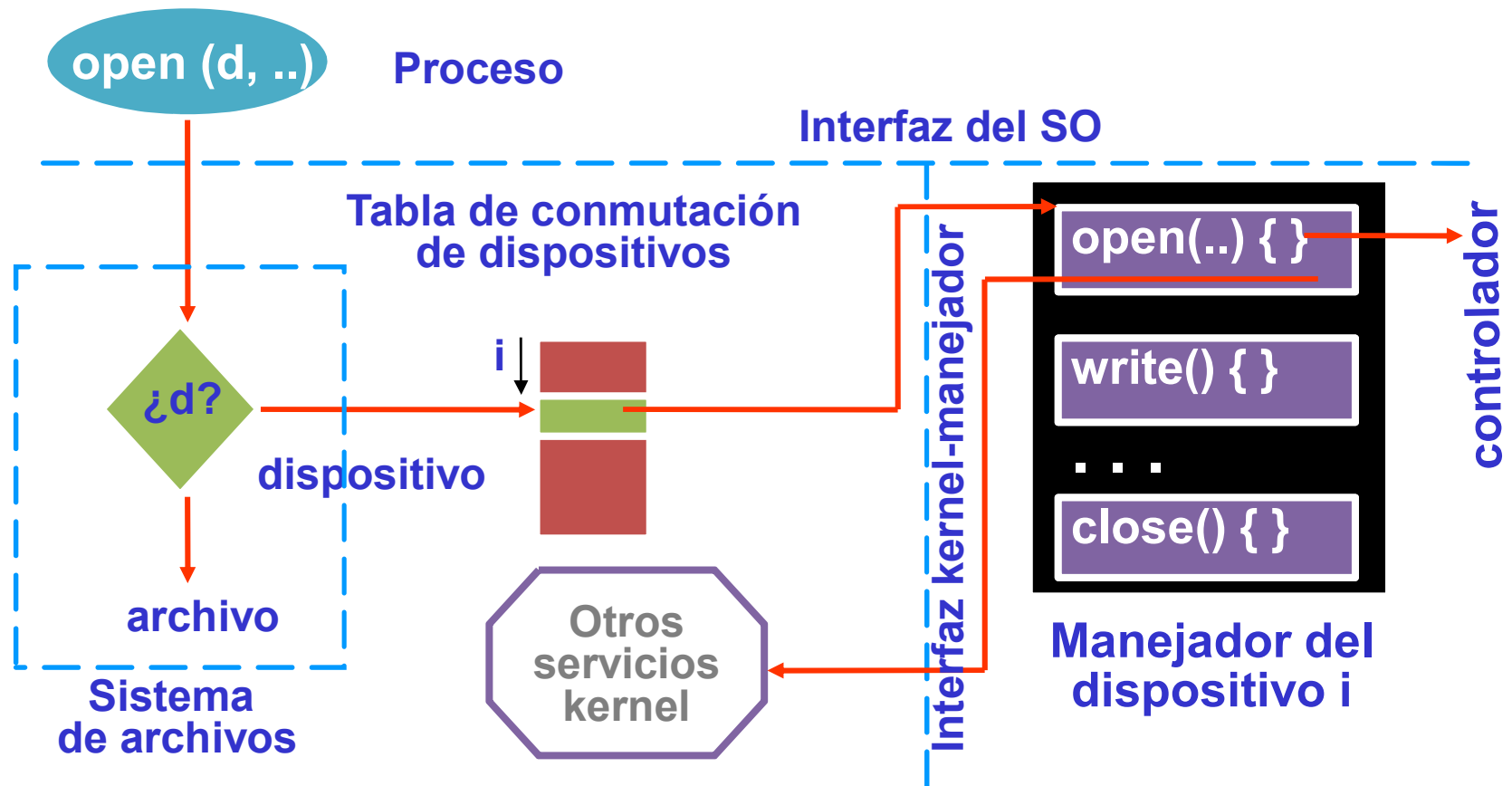
In the terminal output, the number '8' in the first line is circled in red, and the number '2' in the second line is circled in green. In the third line, the permissions 'crw-rw-rw-' are circled in blue.

Espacio de nombres de usuario

- ▷ La forma de designación más extendida es la integración del espacio de nombres de dispositivos en el de archivos.
- ▷ El concepto central es el **archivo de dispositivo**: permite manejar de la misma forma archivos y dispositivos (mismo conjunto de llamadas al sistema), y aplicarles los mismos mecanismo de protección.
- ▷ En UNIX un archivo dispositivo es un archivo especial que no contiene datos (el campo tamaño ha sido sustituido por los números principales y secundarios), es una interfaz al controlador. Ej. Leer del archivo dispositivo terminal se traduce en una llamada a leer directamente del terminal.



Designación e independencia del dispositivo



Búfering

- ▷ Los dispositivos suelen tener una pequeña memoria en la tarjeta para almacenar datos temporalmente antes de transferirlos a/desde la CPU.
- ▷ ¿Por qué tener búferes en el SO? Para:
 - Acoplar la diferencia de velocidades entre la CPU y el dispositivo.
 - Hacer frente a la diferencia de tamaños de transferencia de datos entre dispositivos.
 - Minimizar el tiempo que el proceso de usuario esta bloqueado en una escritura.



Caché de búferes

- ▷ Mejorar el rendimiento del disco reduciendo el número de acceso al disco.
- ▷ Mantener bloques de datos recientemente usados en memoria después de que se complete la llamada de E/S que lo trajo.

```
Ej.:      read(DireccionDisco) {  
          if (bloque en memoria) then  
              Retorna su valor;  
          else
```

```
LeeSector(DireccionDisco);  
};
```



Caché de búferes (y ii)

▷ Ej. sustituido por:

```
write(direcciónDisco) {  
  if (bloque en memoria) then  
    actualiza su valor;  
  else  
    asignar espacio en memoria;  
    leelo de disco;  
    actualiza su valor en memoria; }
```

▷ Políticas de escritura:

- **Write-through** - escribir todos los niveles de memoria que contienen el bloque, incluido el disco. Muy fiable.
- **Write-back** - escribir sólo en memoria más rápida que contiene el bloque; escribir en memorias más lentas y disco más tarde. Más rápido.

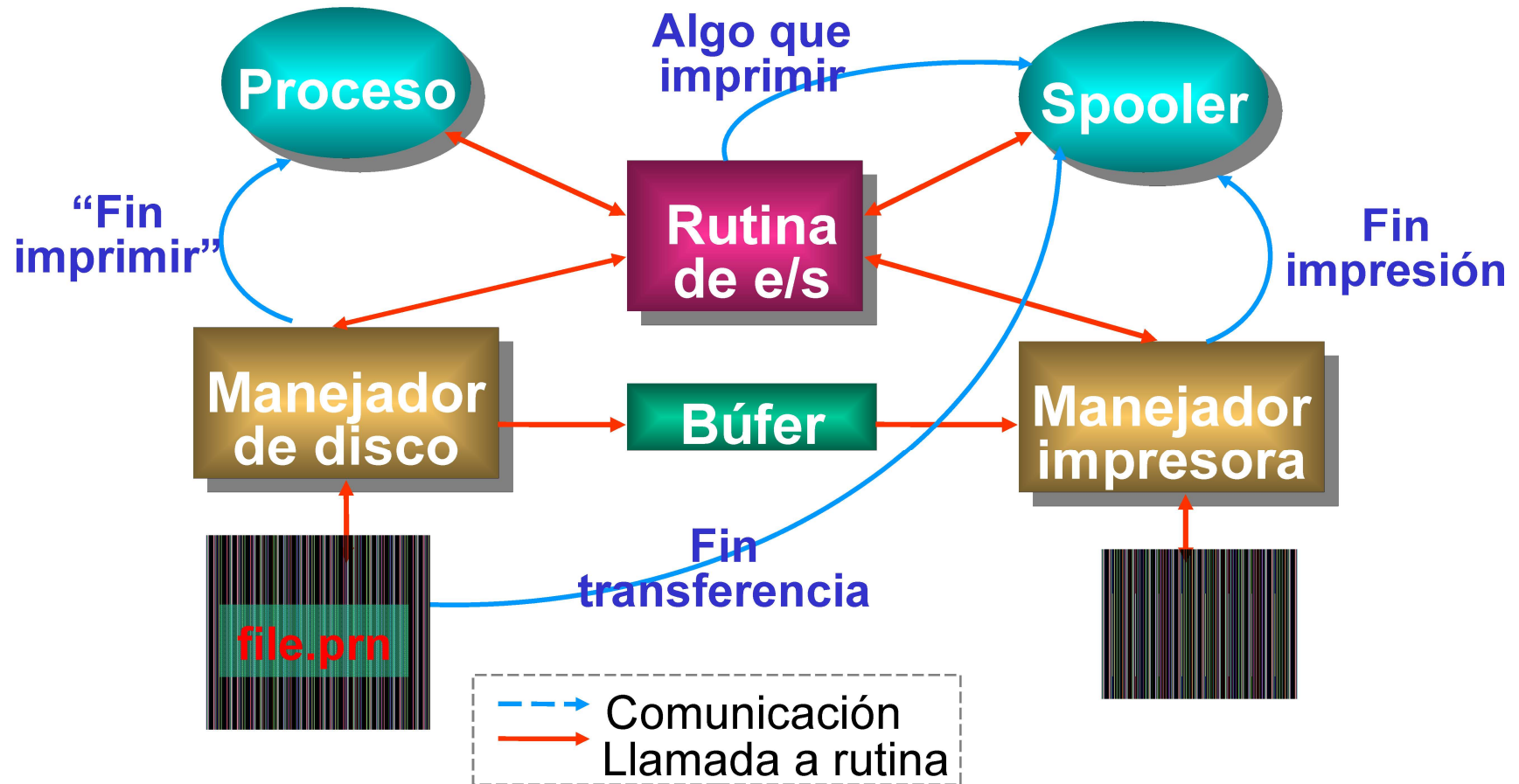


E/S en espacio de usuario

- ▷ *Bibliotecas estándar de e/s* - permiten realizar las llamadas al sistema de e/s para:
 - Gestión de formatos (ej. `printf`)
 - Control de los dispositivos (p. ej. `ioctl`)
- ▷ **Spooling** - técnica para manejar dispositivos dedicados en sistemas multiprogramados. P. ej. No asignamos impresora sino que generamos la impresión en un archivo; un proceso especial manda los archivos a impresión. Igual que el correo electrónico.



Spooling





3.

Manejadores de dispositivos

**Cosideraciones en la construcción de
manejadores de dispositivos**

Visión del programador

- ▷ El SO suministra una interfaz de los dispositivos que simplifica el trabajo del programador:
 - Interfaz estándar para diferentes dispositivos relacionados.
 - El manejador del dispositivo encapsula las dependencias del dispositivo.
 - El SO puede soportar nuevos dispositivos simplemente con suministrar el manejador del dispositivo.



Visión del programador (ii)

▷ Características del dispositivo:

- Unidad de transferencia: carácter/bloque
- Método de acceso: secuencial/aleatorio
- Temporización: síncrona/asíncrona. Observar que la mayoría de los dispositivos son asíncronos, mientras que la llamadas al sistema de E/S son síncronas. El SO implementa E/S bloqueantes.
- Compartido o dedicado.
- Velocidad.
- Operación: entrada, salida, o ambas.

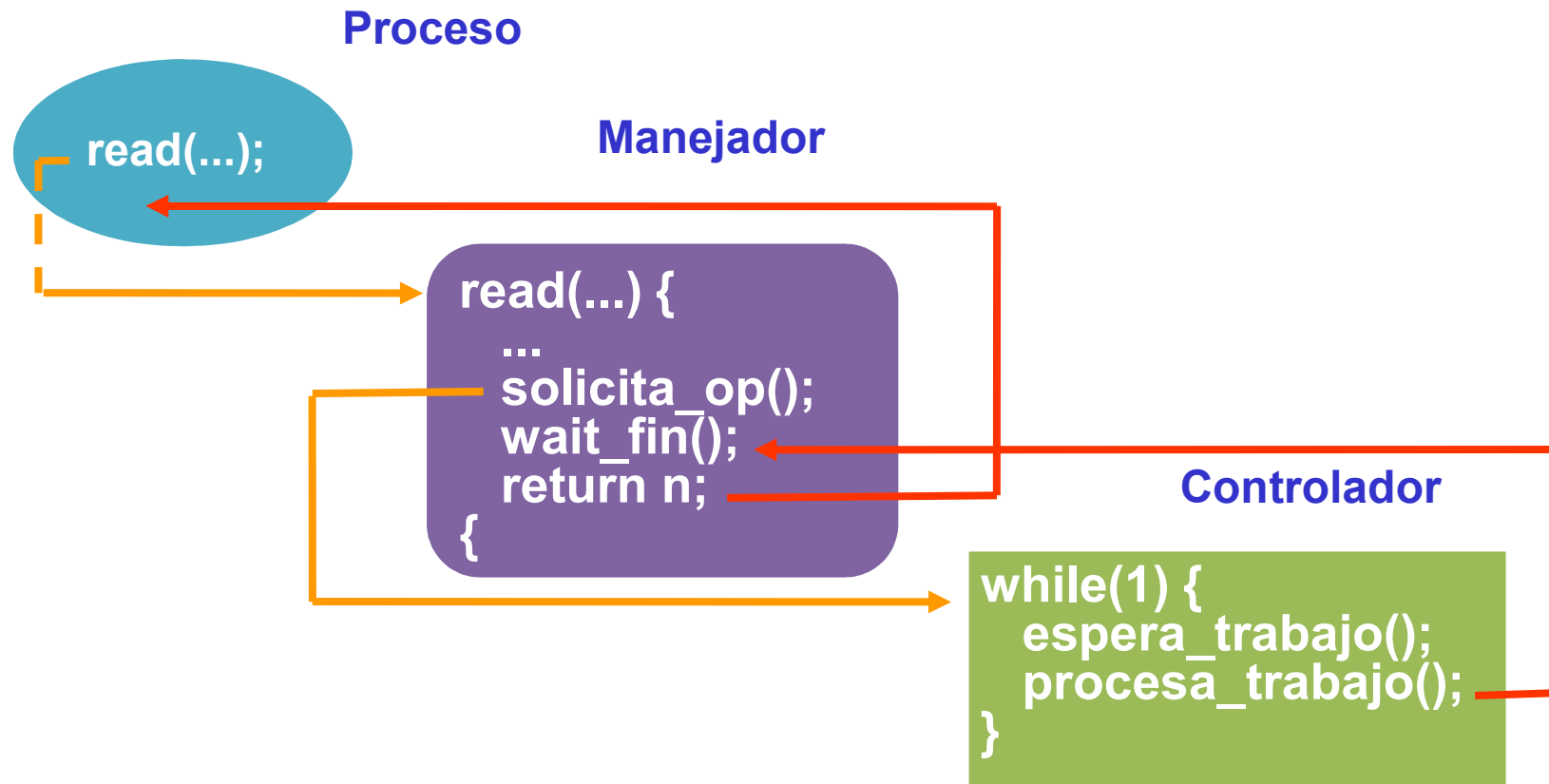


Todo junto: ejemplo de lectura

- ▷ El usuario solicita una lectura de dispositivo
- ▷ Si los datos están en un búfer, salta a paso 3. Si no están:
 - El SO instruye al dispositivo y “espera”.
 - Cuando el dispositivo tiene los datos interrumpe a la CPU, que los transfiere a un búfer del SO. Con DMA ...
- ▷ El SO transfiere los datos al proceso de usuario y lo desbloquea.
- ▷ Cuando el proceso alcanza la CPU, sigue la ejecución después de la llamada.



Lectura: esquema



Problema: un esclavo con dos amos

- ▷ El dibujo anterior muestra como el manejador obedece, por una parte, órdenes del resto de módulos del SO, y, de otra, del controlador del dispositivo.
- ▷ Siendo esto así ¿cómo realiza la espera, `wait_fin()`, el manejador en el paradigma síncrono de programación?
 - El *paradigma síncrono* de programación, el más usado, supone que una vez que la operación de e/s se ha realizado, los datos están listos para usarse. Esto supone que debemos bloquear al proceso en su contexto mientras la operación de e/s finaliza.
 - Por otro lado, la invocación de las rutinas de servicio de interrupción es asíncrona al proceso que espera y, por tanto, no debe utilizar el contexto del proceso para su ejecución.



Solución: Manejador de 2 niveles

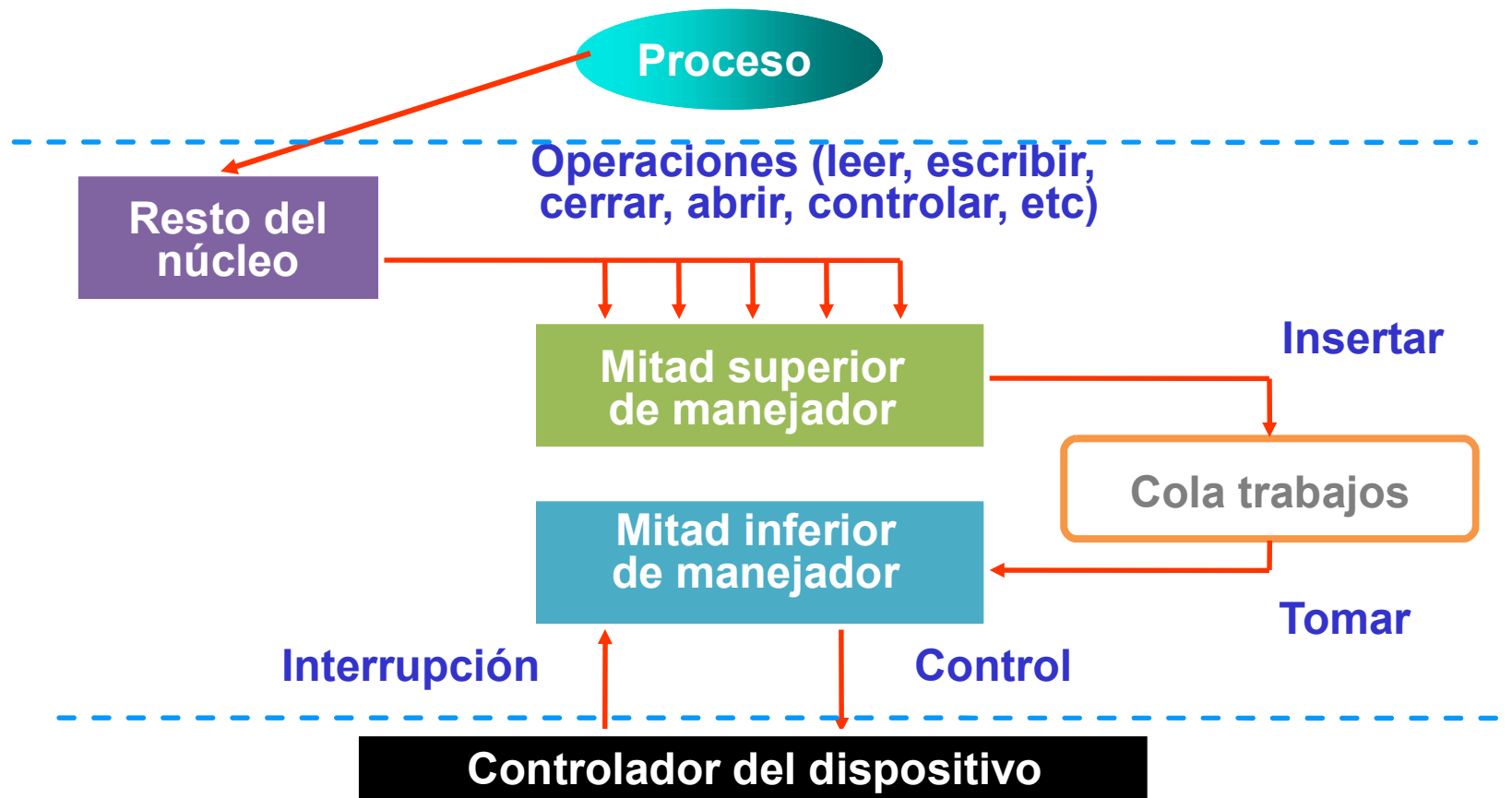
▷ El manejador se divide en dos mitades:

- La **mitad superior** contiene las rutinas síncronas, se ejecuta en el contexto del proceso y puede acceder a su espacio de direcciones. Puede bloquear al proceso.
- La **mitad inferior** contiene las rutinas asíncronas, no accede al contexto del proceso y normalmente no tiene relación con el proceso. No puede bloquearse, pues bloquearía a un proceso no relacionado con la interrupción.



Manejador: estructura a 2 niveles

(ii)



Rendimiento de E/S

- ▷ Las E/S son costosas por varias razones:
- En disco duros involucran movimientos físicos lentos (cabezal disco) o retardo por líneas de comunicaciones (teléfono-red).
 - Los dispositivos de E/S son a menudo disputados por múltiples procesos.
 - Las operaciones de E/S se suministran por medio de llamadas al sistema y gestión de interrupciones, que son lentas.



Rendimiento de E/S: soluciones

- ▷ **Planificación de disco:** las peticiones para leer/escribir bloques de discos se pueden reorganizar en la cola de trabajos para hacer más eficiente el acceso al dispositivo (reducir la latencia de acceso al mismo).
- ▷ Reducir el número de veces que los datos son copiados manteniéndolos en caché.
- ▷ Reducir la frecuencia de interrupciones utilizando, si es posible, grandes transferencias de datos.
- ▷ Descargar computación de la CPU principal utilizando controladores DMA.
- ▷ Aumentar el número de dispositivos para reducir la contención de uno único, y así, mejorar el uso de CPU.
- ▷ Incrementar memoria física para reducir la cantidad de tiempo en paginación y por ello mejorar el uso de CPU.

