Sistema de Entrada/Salida

David Alejandro González Márquez
Pablo Dobrusin

Tecnología Digital II: Sistemas de Computación
Licenciatura en Tecnología Digital
Universidad Torcuato Di Tella

Clase disponible en: `https://github.com/fokerman/computingSystemsCourse`

Hardware de Entrada/Salida

Anteriormente estudiamos el hardware de entrada/salida.

Tanto **formas de acceder** a dispositivos de E/S.

- Mapeado a memoria
- Puertos de entrada-salida

Como **estrategias de control** de dispositivos.

- Espera activa o *Polling*
- Interrupciones
- Acceso Directo a Memoria (DMA)

Ahora vamos a entrar en detalle sobre cada uno de estos puntos.

Formas de acceso a entrada/salida

Acceso al espacio de memoria de los dispositivos.

- **E/S mapeado a memoria**: Los accesos a memoria son capturados y de estar en el rango de memoria de **los dispositivos, estos son los que responden**, tanto a escrituras como lecturas sobre memoria. La direcciones a las que se accede son **direcciones físicas** que ignoran el mecanismo de paginación. Además, **la memoria cache no puede interferir** en estos accesos, siendo desactivada en el rango de E/S.
- **Puertos de E/S**: Existe un **espacio de direccionamiento independiente** para E/S. A este espacio se accede por un bus independiente. Suele tener **una frecuencia más lenta** que el bus de memoria y un rango de direcciones de menor tamaño.

Para controlar los dispositivos se cuenta con registros especiales que se clasifica como:

- `data-in register`: Se lee por le sistema para obtener datos.
- `data-out register`: El sistema lo escribe para enviar datos.
- `status register`: Contiene bits que se interpretan como el estado del dispositivo.
  Ej. Bit de ocupado, o disponible, bit indicador de finalización de una acción.
- `control register`: Es escrito por el sistema para controlar al dispositivo.
  Ej. Bit de comienzo de acción, bits de selección de velocidad, bit de modo.

Espera activa o *Polling*

El protocolo de espera activa consiste en consultar el estado del dispositivo todo el tiempo hasta que se cumpla una concición que permita continuar.

Es un mecanismo eficiente, mientras la espera no demore mucho.

Las rutinas que coordinan una espera activa suelen respetar el siguiente protocolo:

1. Se espera que el bit de *busy* del registro de `status` del dispositivo, este libre.
2. El sistema escriben datos en los registros de `data-out` para cargar la información a utilizar.
3. El sistema setea en el registro de `control` el bit de *ready*.
4. El dispositivo detecta el bit *ready* y utiliza la información los registros `data-out` para continuar.
5. El dispositivo escribe el registro de `status` indicando que termino y que no hay errores.

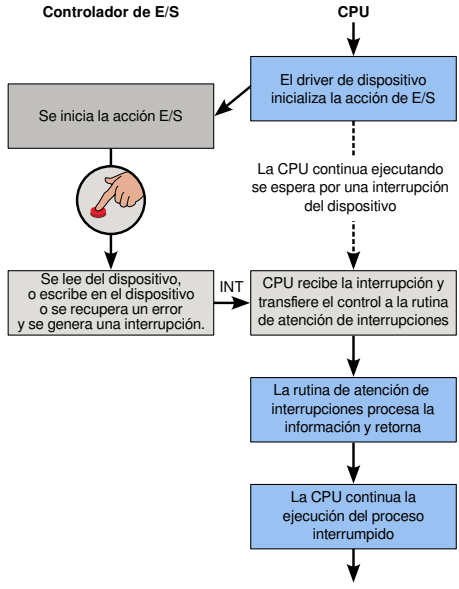
Interrupciones

El mecanismo de interrupciones cuenta con **rutinas** que son ejecutadas cuando se presenta una interrupción.

Las rutinas pueden atender a **múltiples dispositivos** o a uno solo, además son atentidas según su **prioriodad**. Incluso, pueden estar **enmascaradas** impidiendo que puedan interrumpir la ejecución de otra rutina.

Desde el punto de vista del sistema operativo se suele utilizar el siguiente protocolo:

1. El sistema mediante el driver, configura al dispositivo para realizar una operación de E/S.
2. El dispositivo inicia la acción y en algún momento cuando termina, genera una interrupción.
3. La interrupción es atendida por el procesador, este recibe o genera la información del dispositivo y continua la ejecución del proceso interumpido.



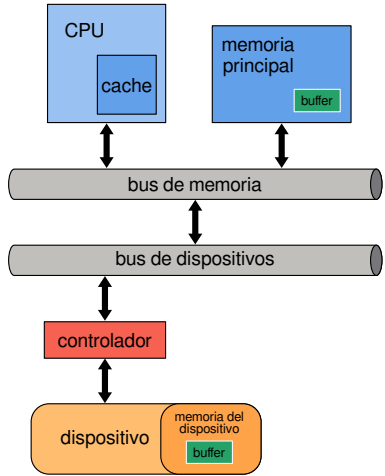
Acceso Directo a Memoria (DMA)

Los mecanismos de DMA o Entrada/salida programada (PIO) se ocupan de tomar el control del bus de memoria y realizar las tareas de transferencia de datos que realizaría el procesador.

Para esto se debe contar con un *buffer* en memoria física que utilizará el sistema para dejar los datos que el DMA requiera.

El sistema cuenta con multiples dispositivos DMA, debe coordinar su uso, priorizando las tareas de trasnferencia de datos a realizar.

1. El sistema selecciona una tarea de transferencia en la cola de espera del dispositivo DMA.
2. Configura los registros de control y configuración del DMA.
3. Queda a la espera de la interrupción de finalización para cargar una nueva tarea de transferencia.



Tipos de dispositivos de E/S

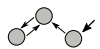
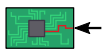
Los dispositivos de E/S son variados, y por lo tanto existen diferentes formas de clasificarlos.

Modo de transferencia de datos	**Flujo de caracteres**: transfiere de a bytes (Ej. teclado) **Flujo de bloques**: transfiere de a bloques de bytes (Ej. disco)
Método de acceso	**Acceso secuencial**: acceso es determinado por el dispositivo (Ej. placa de red) **Acceso aleatorio**: acceso es determinado por el usuario del dispositivo (Ej. disco)
Forma de transferencia	**Síncrona**: el tiempo de respuesta es predecible (Ej. disco óptico) **Asíncrona**: el tiempo de respuesta es aleatorio (Ej. mouse)
Compartición	**Compartible**: varios procesos lo pueden usar concurrentemente (Ej. teclado) **Dedicado**: solo un proceso lo puede usar a la vez (Ej. impresora)
Velocidad	**Latencia**, **tiempo de búsqueda**, **tasa de transferencia**, etc. (varía según cada dispositivos)
Tipo de operación	**Solo lectura** (Ej. DVD **Solo escritura**: (Ej. controladora gráfica) **Lectura-escritura**: (Ej. disco)

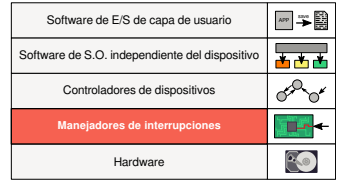
Subsistema de E/S

Cada proceso debe poder comunicarse con los dispositivos que necesita a través del sistema.

Para esto el sistema provee una interfaz que permite abstraer el acceso a los dispositivos en comandos simples, mientras **oculta el funcionamiento interno** de cada dispositivo.

Capa 4		Software de E/S de capa de usuario
Capa 3		Software de S.O. independiente del dispositivo
Capa 2		Controladores de dispositivos
Capa 1		Manejadores de interrupciones y rutinas
Capa 0		Hardware

Capa 1: Manejador de interrupciones y rutinas



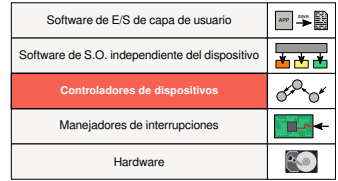
Por medio de un conjunto de rutinas básicas, el sistema operativo puede **atender interrupciones** y dar el control a las capas superiores.

Además utiliza rutinas especiales, que permiten acceder al **espacio de memoria de los dispositivos**.

El sistema cuenta con todo tipo de rutinas para atender dispositivos, desde **rutinas genéricas** para dispositivos, donde solo se provee una interfaz para acceder a memoria, hasta complejas rutinas para **atender dispositivos específicos**.

El desarrollo de esta capa es fundamental para construir drivers de dispositivo eficientes.

Capa 2: Controladores de dispositivos (*drivers*)



El funcionamiento de cada dispositivo depende del fabricante. Este implementa un **protocolo de comunicación particular**.

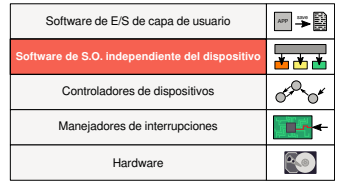
Para esto, los fabricantes proveen ***drivers*** específicos para cada dispositivo y sistema donde serán utilizados.

La interfaz de la capa anterior es utilizada por los *drivers* para acceder directamente al hardware.

Por esta razón, muchas veces los sistemas no implementan las interfaces que necesitan los drivers, y por lo tanto son incompatibles con el sistema. Otras veces, los fabricantes directamente no dan soporte para algunos sistemas operativos.

Los *drivers* deben ser rutinas **reentrantes**, es decir que deben poder recibir una nueva solicitud mientras están ejecutando otra.

Capa 3: Subsistema de E/S independiente del dispositivo



Publica una **interfaz uniforme** para acceder a los dispositivos.

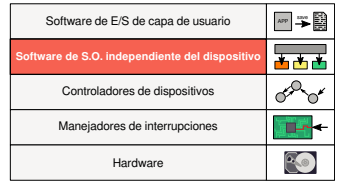
Los *drivers* se deben adaptar y exponer la interfaz que el sistema propone.
De esta forma todos los dispositivos de comportamiento similar,
son utilizandos por los usuarios de la misma forma.
Ej. disco óptico, disco rígido, disco de estado sólido

Además se ocupa de la **Planificación de E/S**.

Determinando el orden en que se atenderán las solicitudes
y respetando las prioridades entre los diferentes dispositivos.

Esta capa también se ocupa de resolver las **operaciones comunes** a todos los dispositivos

Capa 3: Subsistema de E/S independiente del dispositivo



Algunas de las operaciones comunes que resuelve esta capa:

Administración de búferes

Los datos a transferir se almacenan temporalmente en memoria, permitiendo adaptar los tamaños de transferencia entre dispositivos y sistema.

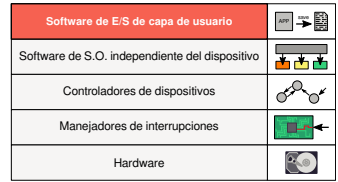
Administración de errores

Los errores son capturados poer el sistema como resultado de una operación de E/S. Existen errores que pueden dejar al sistema en un estado inconsistente que debe ser administrado correctamente para evitar la perdida de datos.

Asignación y liberación de dispositivos dedicados

El sistema debe aceptar y rechazar solicitudes de uso de dispositivos teniendo en cuenta las características de uso de cada uno. Evitando en todo momento generar *deadlocks* por condiciones de sincronización.

Capa 4: Software de E/S en la capa de usuario



Esta capa esta formada por el conjunto de **bibliotecas de funciones** que resuelven el acceso a E/S:

Proporcionan comandos o funciones que nos permiten utilizar los recursos que proveen los dispositivos de forma más simple.

Por ejemplo, la función `printf` que se utiliza para imprimir en pantalla, lama internamente a *syscalls* como `write` cuando se debe imprimir un texto largo o `put` cuando se debe imprimir un solo caracter.

Mientras tanto, open y *close* son funciones que se pueden utilizar para abrir y cerrar archivos o dispositivos de caracteres mapeados archivos.

Otro ejemplo es `write` que puede ser utilizada como *syscall* para imprimir en pantalla o para imprimir dentro de un archivo, incluso información binaria.

Relojes

Los relojes son un tipo específico de dipositivo de entrada/salida.

Su interfaz de hardware se ocupa de **generar eventos en intervalos regulares de tiempo**.
Estos eventos se suelen traducir en interrupciones o dejan registro en contadores.

Los sistemas cuentan con múltiples relojes que operan a distintas frecuencias

El sistema distingue entre distintas fuentes de tiempo:

- Temporizador de tiempo real (*real-time-clock*, RTC).
- Temporizador de eventos de alta precisión (*high precision event timer*, HPET)

La interfaz del reloj provee tres funciones básicas:

- Obtener el tiempo actual.
- Obtener el tiempo transcurrido desde el último evento.
- Configurar el reloj para despertar la operación X en el momento T.

Internamente el sistema requiere múltiples relojes, para resolver esto, se crean **relojes virtuales**.
Los relojes virtuales se actualizan o cambian su estado en función de los relojes físicos.

Administración de energía

El sistema operativo juega un rol fundamental en la administración de la energía.

Ya sea que se cuente con una bateria como fuente de alimentación o se esté conectado a la red eléctrica. La **reducción del consumo** de energía es un eje básico de diseño.

Los sistemas están diseñados para disipar una determinada cantidad de energía que se manifiesta en **calor**. Este calor debe ser reducido mediante sistemas de enfriamiento como *coolers*.

Advanced Configuration and Power Interface (ACPI)

Provee un estándar para que el sistema operativo pueda descubrir y configurar los componentes de hardware del sistema y administrar el consumo de energía. Permite monitorar el uso del los componentes, su estado, y mecanismos de *Plug and Play* y *hot swapping*.

El sistema operativo se encarga de decidir cuando **encender o apagar componentes**, o incluso **degradar su funcionamiento** para reducir el consumo.

Esta tarea es muy compleja ya que para reducir el consumo se debe **afectar la experiencia de usuario.**

Bibliografía

- Silberschatz, “Fundamentos de Sistemas Operativos”, 7ma Edición, 2006.
  - **Capítulo 13 - Sistema de E/S**
    - 13.3 Interfaz de E/S de las aplicaciones
    - 13.4 Subsistema de E/S del kernel
    - 13.5 Transformación de las solicitudes de E/S en operaciones hardware

- Tanenbaum, “Modern Operating Systems”, 4th Edition, 2015.
  - **Chapter 5 - Input/Output**
    - 5.2 Principles of I/O software
    - 5.3 I/O software layers
    - 5.5 Clocks
    - 5.8 Power management

¡Gracias!