Resumen ISO → Tema IO(entrada/salida)

Responsabilidades del SO

- Controlar dispositivos de E/S
  - Generar comandos.
  - Manejar interrupciones.
  - Manejar errores.
- Proporcionar una interfaz para utilizar los dispositivos de E/S

Problemas en E/S

- Dispositivos diferentes entre sí.
- Características únicas de cada uno.
- Velocidad de cada dispositivo (variante).
- Surgen nuevos tipos de dispositivos que el SO debe adaptar para utilizarlos (siempre y cuando no se haga un cambio muy fuerte del diseño del SO).
- Existen diferentes formas de realizar E/S (ver anexo).

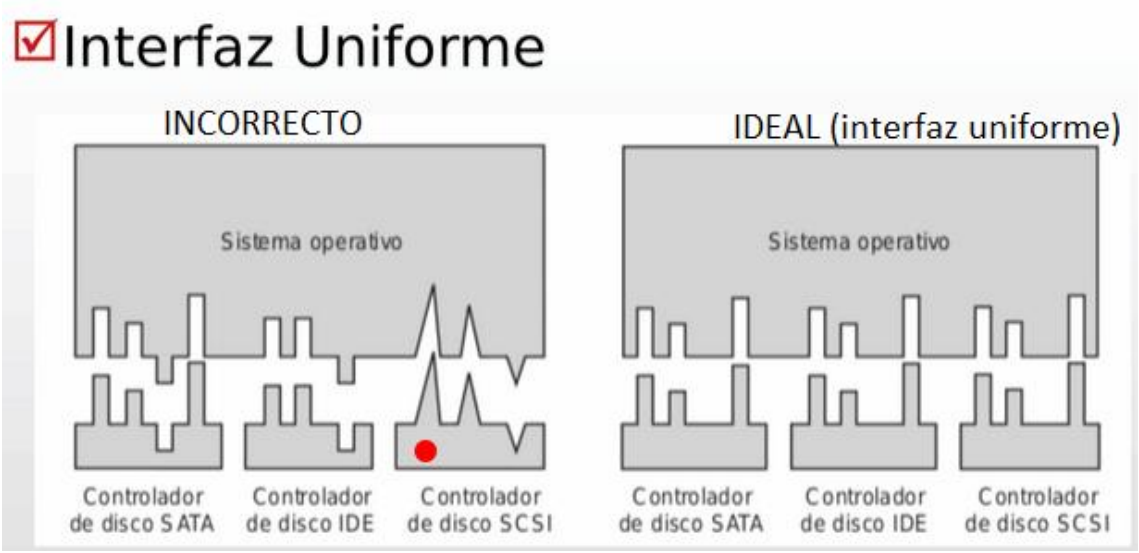
Aspectos de los dispositivos de E/S

- Unidad de transferencia:
  - Dispositivos que transfieren bloques (discos):
    - Operaciones → Read, Write, Seek
  - Dispositivos que transfieren caracteres (teclados, mouse, serial ports):
    - Operaciones → Get, Put.
- Forma de acceder al dispositivo:
  - Secuencialmente.
  - Aleatoriamente.
- Tipo de acceso al dispositivo:
  - Acceso compartido (más de un proceso usando el mismo dispositivo): Disco rígido.
  - Acceso exclusivo (un solo proceso por vez): Impresora.
- Tipo de operaciones:
  - Read only: CDROM.
  - Write only: Pantalla.
  - Read/write: Disco.

Metas, objetivos y servicios

- Generalidad:
  - Es deseable manejar todos los dispositivos de una manera estandarizada (general).

- Se deben ocultar la mayoría de los detalles del dispositivo en las rutinas de niveles más “bajos” de forma que los procesos vean a los dispositivos en forma de operaciones comunes → read, write, open, close, lock, unlock.

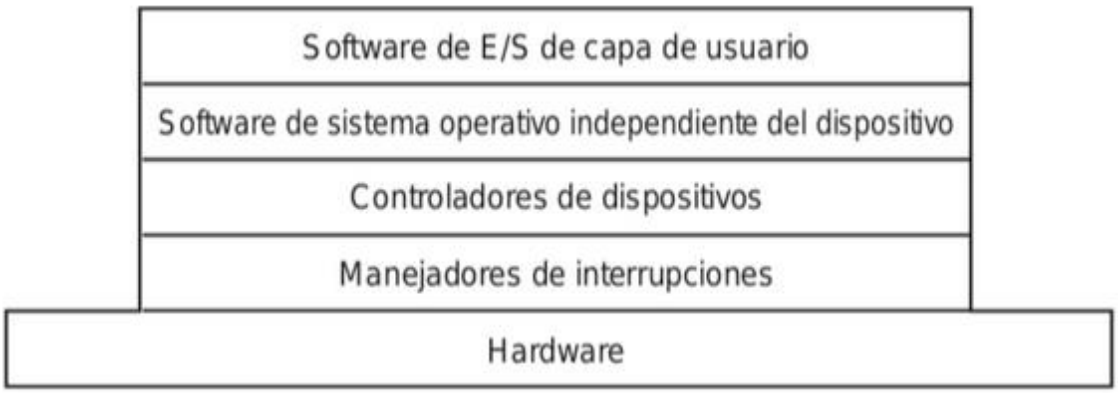


- Eficiencia:
  - Los dispositivos de E/S son mucho más lentos respecto a la memoria RAM y la CPU.
  - Aprovechando la idea de la multi-programación permite que un proceso espere por la finalización de su E/S mientras que otro proceso se ejecuta (no desperdiciamos uso de CPU)
- Planificación:
  - Se deben organizar los requisitos de los dispositivos.
  - Ejemplo: planificar requerimientos a disco para minimizar tiempos.
- Buffering: almacenamiento de los datos en memoria mientras se transfieren.
  - Soluciona problemas de velocidad entre todos los dispositivos.
  - Soluciona problemas de tamaño y/o forma de los datos entre los dispositivos.
- Caching: se mantiene en memoria una copia de los datos de reciente acceso para mejorar la performance.
- Spooling: administrar la cola de requerimientos de un dispositivo.
  - Mecanismo que arma una cola para coordinar el acceso concurrente a un dispositivo.
  - Esto se usa ya que algunos dispositivos de acceso **exclusivo,** no pueden atender distintos requerimientos al mismo tiempo. Por ej: impresora.
- Reserva de dispositivos: para dispositivos de acceso exclusivo.

Resumen ISO → Tema IO(entrada/salida)

- Manejo de errores:
  - El S.O administra los errores ocurridos(lectura de un disco, dispositivo no disponible, errores de escritura).
  - La mayoría retorna un número de error o código cuando la I/O falla.
  - Logs de errores.
- Formas de realizar E/S:
  - Bloqueante: el proceso es suspendido hasta que el requerimiento de E/S es completado.
    - Fácil de usar y entender.
    - No es suficiente bajo algunas necesidades.
  - No bloqueante: el requerimiento de E/S retorna en cuanto es posible (el proceso sigue ejecutandose).
    - Ejemplo: aplicación de video que lee frames desde un archivo mientras va mostrándolo en pantalla.

Diseño del software: (diseño de programación en capas, cada capa tiene objetivos muy específicos. La implementación de cada capa es para sí misma → se ocultan detalles de implementación)



Software de capa de usuario (la única en modo usuario)

- Tiene un conjunto de librerías de funciones que no requieren ejecución en modo kernel.
  - Permiten acceso a SysCalls.
  - Implementan servicios que no dependen del kernel.
- Procesos de apoyo.
  - Demonio de impresión (realiza el spooling): la cola se maneja por un proceso en **modo usuario**.

Software de SO independiente del dispositivo (ya en modo kernel)

Resumen ISO → Tema IO(entrada/salida)

- Brinda los principales servicios de E/S antes vistos.
    - Interfaz uniforme.
    - Manejos de errores.
    - Buffering de datos.
    - Asignación de recursos.
    - Planificación.
- Es la capa independiente de los dispositivos → brinda la generalidad.
    - Explotable en sub-capas para realizar un diseño más modular.
- El kernel debe mantener la información de estado de cada dispositivo o componente.
    - Archivos abiertos.
    - Conexiones de red.
    - Etc.
- Hay varias estructuras complejas que representan buffers, utilización de la memoria, disco, etc.

Controladores de dispositivos (drivers)

- Los drivers contienen el código dependiente del dispositivo.
- Manejan un tipo dispositivo.
- Traducen los requerimientos abstractos recibidos de capas más altas en los comandos necesarios para el dispositivo.
    - Escribe sobre los registros del controlador.
    - Acceso a la memoria mapeada.
    - Encola requerimientos.
- Las interrupciones generadas por los dispositivos son atendidas por funciones provistas por el driver.
- En resumen, es el que sabe como manejar el dispositivo.
- Constituye una interfaz entre el SO y el Hardware.
- Los drivers forman parte del espacio de memoria del kernel.
    - Se cargan como módulos.
- Los fabricantes de HW implementan el driver en función de una API especificada por el SO.
    - Open(),close(), read(), write(), etc.
- Para agregar nuevo HW sólo basta indicar el driver correspondiente sin necesidad de cambios en el kernel.
    - El kernel es modular → es decir tiene su parte principal y se le puede agregar módulos (drivers).
- Ejemplo en **LINUX:**

Resumen ISO → Tema IO(entrada/salida)

- ☑ Linux distingue 3 tipos de dispositivos
  - ✓ Carácter: I/O programada o por interrupciones
  - ✓ Bloque: DMA
  - ✓ Red: Ports de comunicaciones
- ☑ Los Drivers se implementan como módulos
  - ✓ Se cargan dinámicamente
- ☑ Debe tener al menos estas operaciones:
  - ✓ init_module: Para instalarlo
  - ✓ cleanup_module: Para desinstalarlo.

- ☑ Operaciones que debe contener para I/O
  - ✓ open: abre el dispositivo
  - ✓ release: cerrar el dispositivo
  - ✓ read: leer bytes del dispositivo
  - ✓ write: escribir bytes en el dispositivo
  - ✓ ioctl: orden de control sobre el dispositivo

- ☑ Otras operaciones menos comunes
  - ✓ llseek: posicionar el puntero de lectura/escritura
  - ✓ flush: volcar los búferes al dispositivo
  - ✓ poll: preguntar si se puede leer o escribir
  - ✓ mmap: mapear el dispositivo en memoria
  - ✓ fsync: sincronizar el dispositivo
  - ✓ fasync: notificación de operación asíncrona
  - ✓ lock: reservar el dispositivo
  - ✓ .....

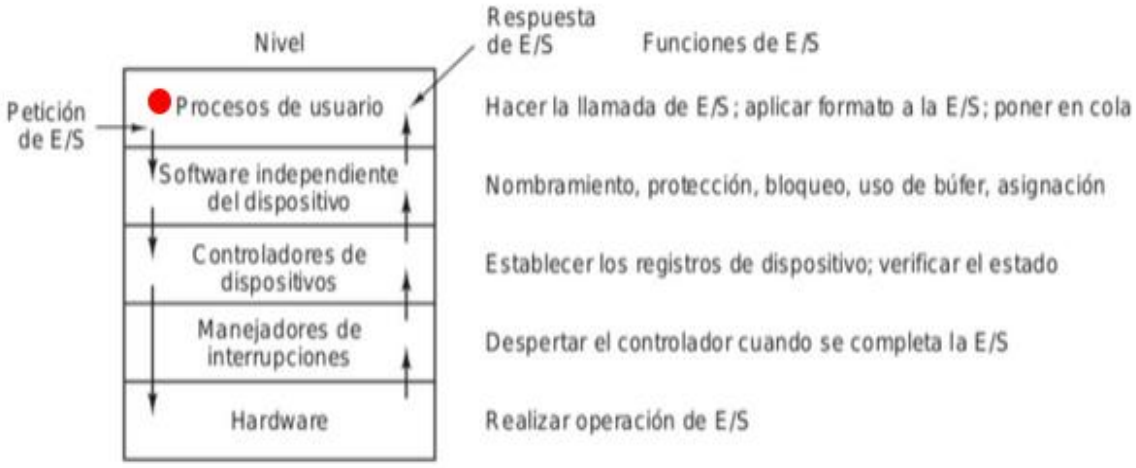
Resumen ISO → Tema IO(entrada/salida)

- ☑ Por convención, los nombres de las operaciones comienzan con el nombre del dispositivo
- ☑ Por ejemplo, para /dev/ptr

```
int ptr_open (struct inode *inode, struct file *filp)

void ptr_release (struct inode *inode, struct file *filp)

ssize_t ptr_read (struct file *flip, char *buff,
                  size_t count, loff_t *offp)

ssize_t ptr_write (struct file *filp, const char *buff,
                   size_t count, loff_t *offp)

int ptr_ioctl (struct inode *inode, struct file *filp,
               unsigned int cmd, unsigned long arg)
```

Manejadores de interrupciones (gestor)

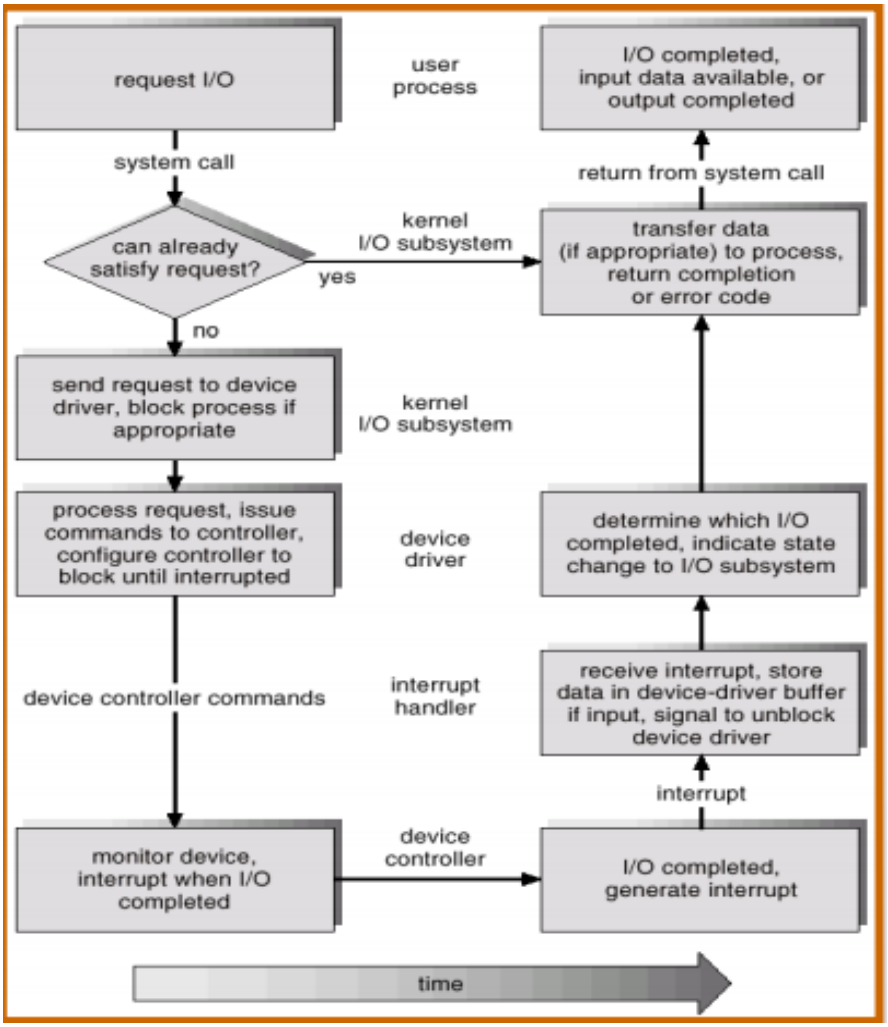
- Atiende todas las interrupciones del HW.
- Deriva al driver correspondiente según interrupción.
- Resguarda información.
- Independiente del driver.
- Puente entre interrupciones y driver que las resuelve.
- Ciclo de atención de un requerimiento:



  - Ciclo explicado:
    - Se determina el dispositivo que almacena los datos.
      - Se traduce el nombre del archivo en la representación del dispositivo.
    - Se traduce el requerimiento abstracto en bloques de discos del filesystem.
    - Se realiza la lectura física de los datos en la memoria.

- Se marcan los datos como disponibles al proceso que realizó el requerimiento.
  - Se desbloquea el proceso.
- Se retorna el control al proceso.

- Ciclo de vida del requerimiento de E/S



Performance

- La I/O es uno de los factores que más afectan a la performance del sistema.
  - Utilizan mucho CPU para ejecutar los drivers y el código del subsistema de I/O.
  - Provoca context switches ante las interrupciones y bloqueos de los procesos.
  - Utiliza el bus de memoria en copia de datos:
    - Aplicaciones (espacio usuario) – kernel.
    - Kernel (memoria fisica) – controlador

Resumen ISO → Tema IO(entrada/salida)

Como mejoro la performance

- Reducir el número de context switches.
- Reducir la cantidad de copias de los datos mientras se pasan del dispositivo a la aplicación (los datos se pasan entre capas).
- Reducir la frecuencia de las interrupciones:
  - Transfiriendo gran cantidad de datos de un saque.
  - Controladores más inteligentes.
  - Polling.
- Utilizar DMA → resuelve las interrupciones, sin tener que usar CPU.