Introducción a los Sistemas Operativos

Subsistema de Entrada / Salida

I.S.O.

- ☑ Versión: Octubre 2018
- ☑ Palabras Claves: Metas, Aspectos de dispositivos, Subsistema de IO

Algunas diapositivas han sido extraídas de las ofrecidas para docentes desde el libro de Stallings (Sistemas Operativos) y el de Silberschatz (Operating Systems Concepts). También se incluyen diapositivas cedidas por Microsoft S.A.

Interfaz de I/O - Metas

- Generalidad:
  - Es deseable manejar todos los dispositivos de I/O de una manera uniforme, estandarizada
  - Ocultar la mayoría de los detalles del dispositivo en las rutinas de niveles más “bajos” para que los procesos vean a los dispositivos, en términos de operaciones comunes como: read, write, open, close, lock, unlock

Interfaz de I/O - Metas

- Eficiencia
  - Los dispositivos de I/O pueden resultar extremadamente lentos respecto a la memoria y la CPU
  - El uso de la multiprogramación permite que un procesos espere por la finalización de su I/O mientras que otro proceso se ejecuta

Aspectos de los dispositivos de I/O

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 readÐwrite	CD-ROM graphics controller disk

Aspectos de los dispositivos de I/O (cont)

- Unidad de Transferencia
  - Dispositivos por bloques (discos):
    - Operaciones: `Read, Write, Seek`
  - Dispositivos por Caracter (keyboards, mouse, serial ports)
    - Operaciones: `get, put`
- Formas de Acceso
  - Secuencial o Aleatorio

Aspectos de los dispositivos de I/O (cont)

- Tipo de acceso
  - Acceso Compartido: Disco Rígido
  - Acceso Exclusivo: Impresora

- Tipo de acceso:
  - Read only: CDROM
  - Write only: Pantalla
  - Read/Write: Disco

Aspectos de los dispositivos de I/O (cont)

- Velocidad

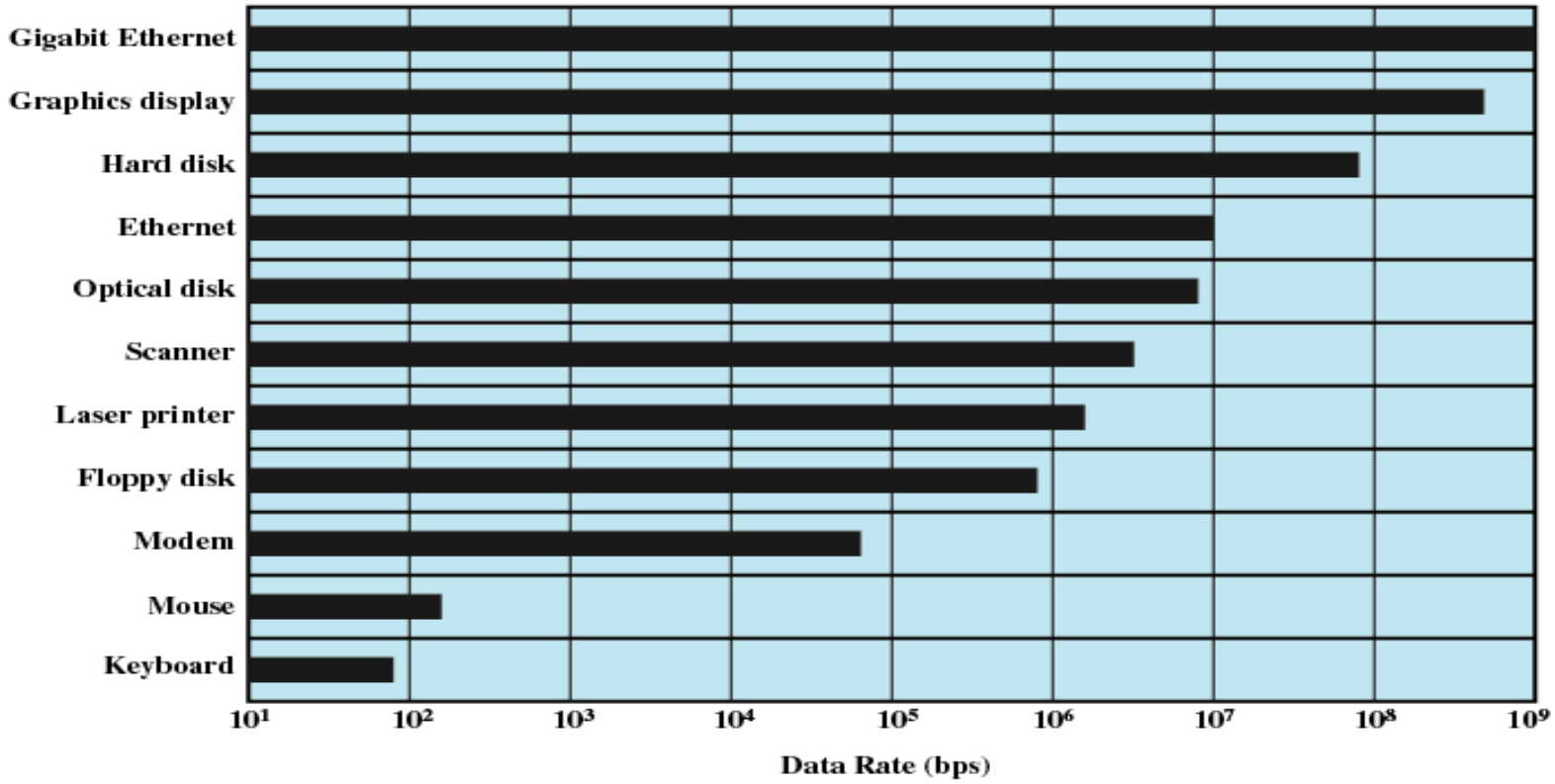
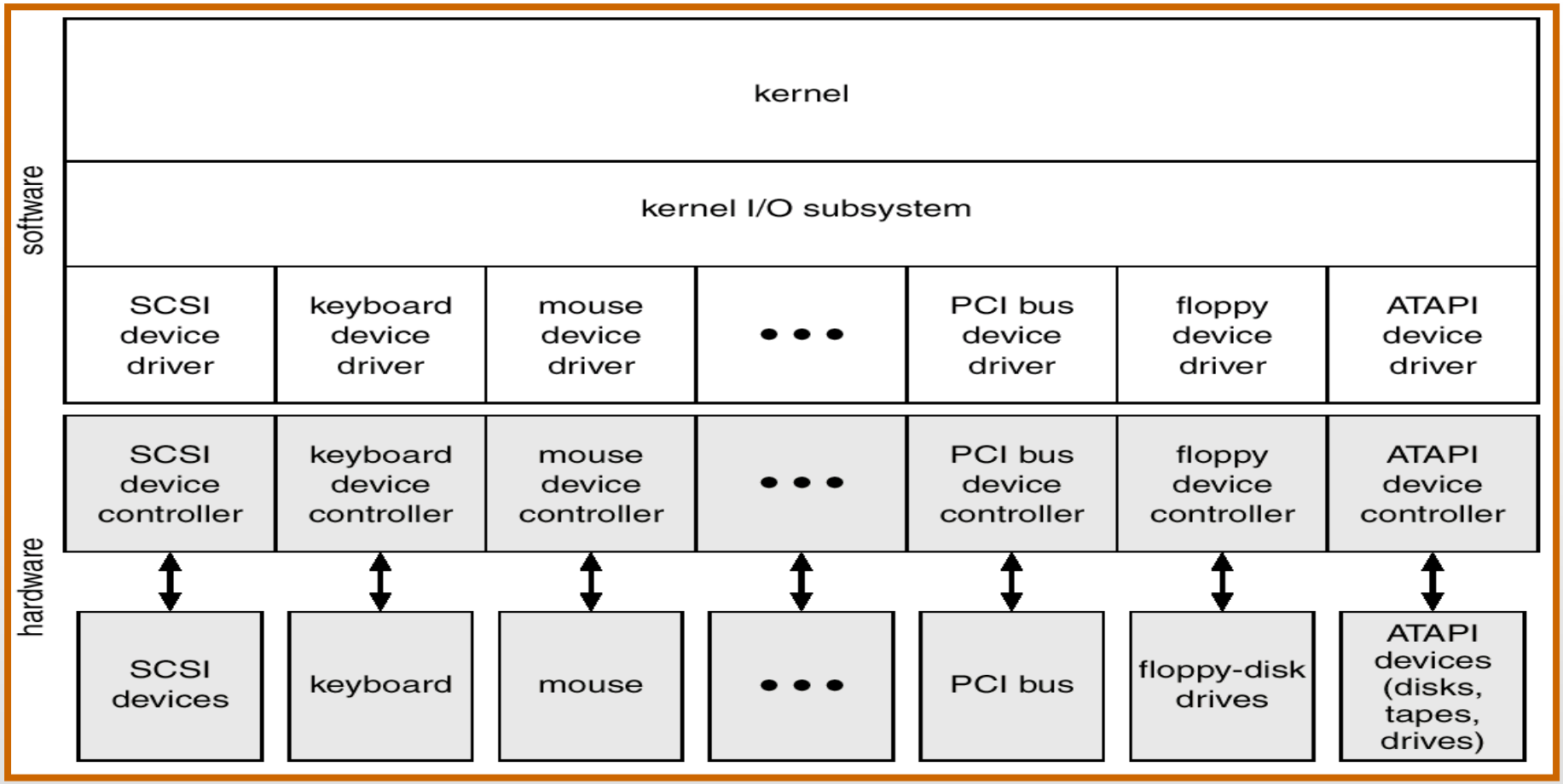


Figure 11.1 Typical I/O Device Data Rates

Subsistema de I/O



Subsistema de I/O - Servicios

- Planificación
  - Organización de los requerimientos a los dispositivos
  - Ej: Planificación de requerimientos a disco para minimizar tiempos
- Buffering – Almacenamiento de los datos en memoria mientras se transfieren
  - Solucionar problemas de velocidad entre los dispositivos
  - Solucionar problemas de tamaño y/o forma de los datos entre los dispositivos

Subsistema de I/O - Servicios (cont.)

- Caching – Mantener en memoria copia de los datos de reciente acceso para mejorar performance

- Spooling – Administrar la cola de requerimientos de un dispositivo
  - Algunos dispositivos de acceso exclusivo, no pueden atender distintos requerimientos al mismo tiempo: Por ej. Impresora
  - Spooling es un mecanismo para coordinar el acceso concurrente al dispositivo

Subsistema de I/O - Servicios (cont.)

- Reserva de Dispositivos: Acceso exclusivo
- Manejo de Errores:
  - El S.O. debe administrar 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

Subsistema de I/O - Servicios (cont.)

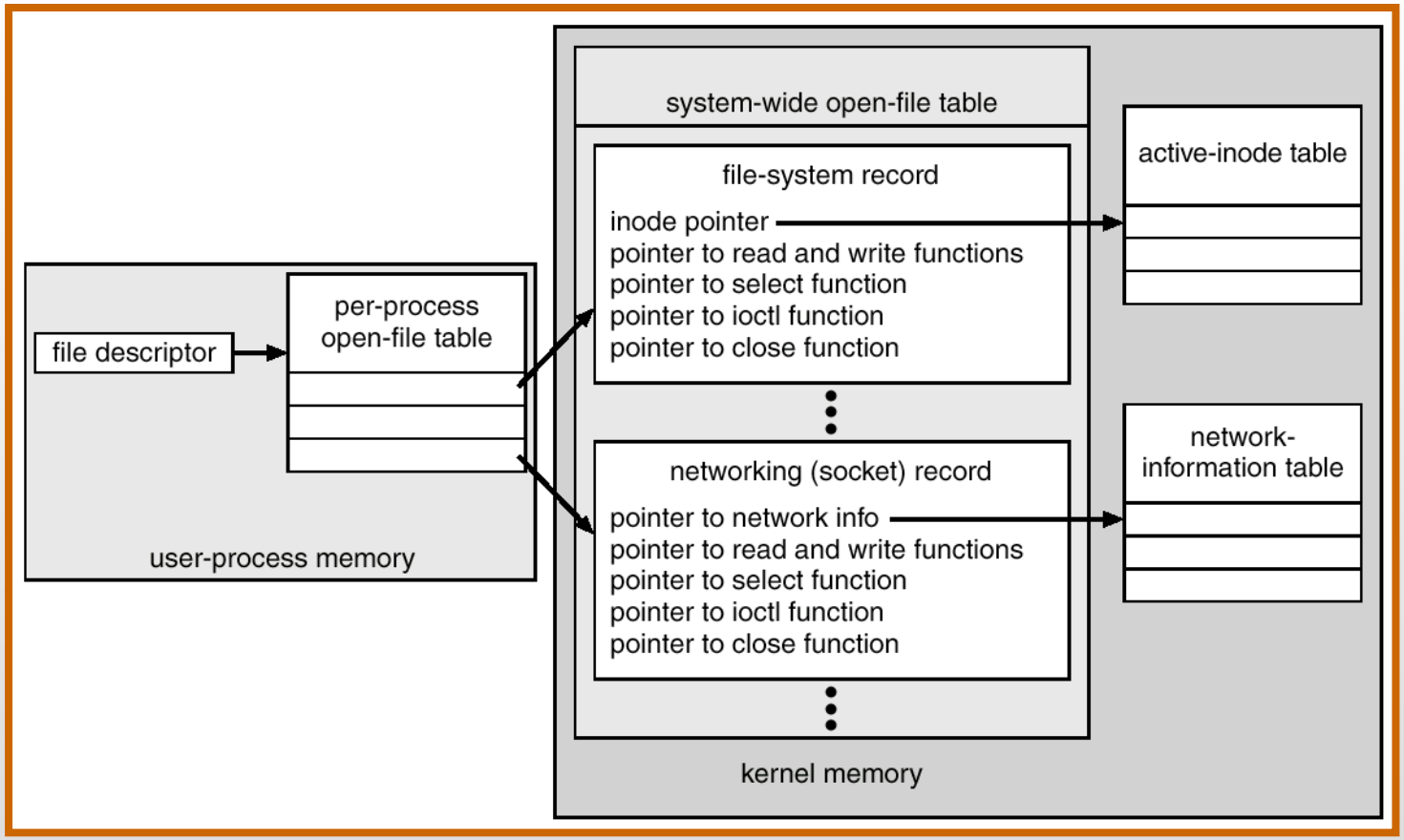
- Formas de realizar I/O
  - Bloqueante: El proceso se suspende hasta que el requerimiento de I/O se completa
    - Fácil de usar y entender
    - No es suficiente bajo algunas necesidades
  - No Bloqueante: El requerimiento de I/O retorna en cuanto es posible
    - Ejemplo: Interfaz de usuario que recibe input desde el teclado/mouse y se muestra en el screen.
    - Ejemplo: Aplicación de video que lee frames desde un archivo mientras va mostrandolo en pantalla.

Subsistema de I/O - Estructuras de Datos

- El Kernel mantiene 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.

Subsistema de I/O - Estructura de Datos

UNIX I/O Kernel Structure



Desde el Requerimiento de I/O hasta el Hardware

- Consideremos la lectura sobre un archivo en un disco:
  - Determinar el dispositivo que almacena los datos
    - Traducir el nombre del archivo en la representación del dispositivo.
  - Traducir requerimiento abstracto en bloques de disco (Filesystem)
  - Realizar la lectura física de los datos (bloques) en la memoria
  - Marcar los datos como disponibles al proceso que realizo el requerimiento
    - Desbloquearlo
  - Retornar el control al proceso

Desde el Requerimiento de I/O hasta el Hardware

```
nico@yoko:~$ ls -l /dev/sd*
brw-rw---- 1 root disk 8,  0 oct 28 11:32 /dev/sda
brw-rw---- 1 root disk 8,  1 oct 28 11:32 /dev/sda1
brw-rw---- 1 root disk 8,  2 oct 28 11:32 /dev/sda2
brw-rw---- 1 root disk 8,  5 oct 28 11:32 /dev/sda5
brw-rw---- 1 root disk 8, 16 oct 28 15:49 /dev/sdb
brw-rw---- 1 root disk 8, 17 oct 28 15:49 /dev/sdb1
nico@yoko:~$
```

```
nico@yoko:~$ cat /proc/devices
Character devices:
  1 mem
  4 /dev/vc/0
  4 tty
  4 ttyS
  5 /dev/tty
  5 /dev/console
  5 /dev/ptmx
  5 ttyprintk
```

```
Block devices:
  1 ramdisk
259 blkext
  7 loop
  8 sd
  9 md
 11 sr
 65 sd
 66 sd
 67 sd
```

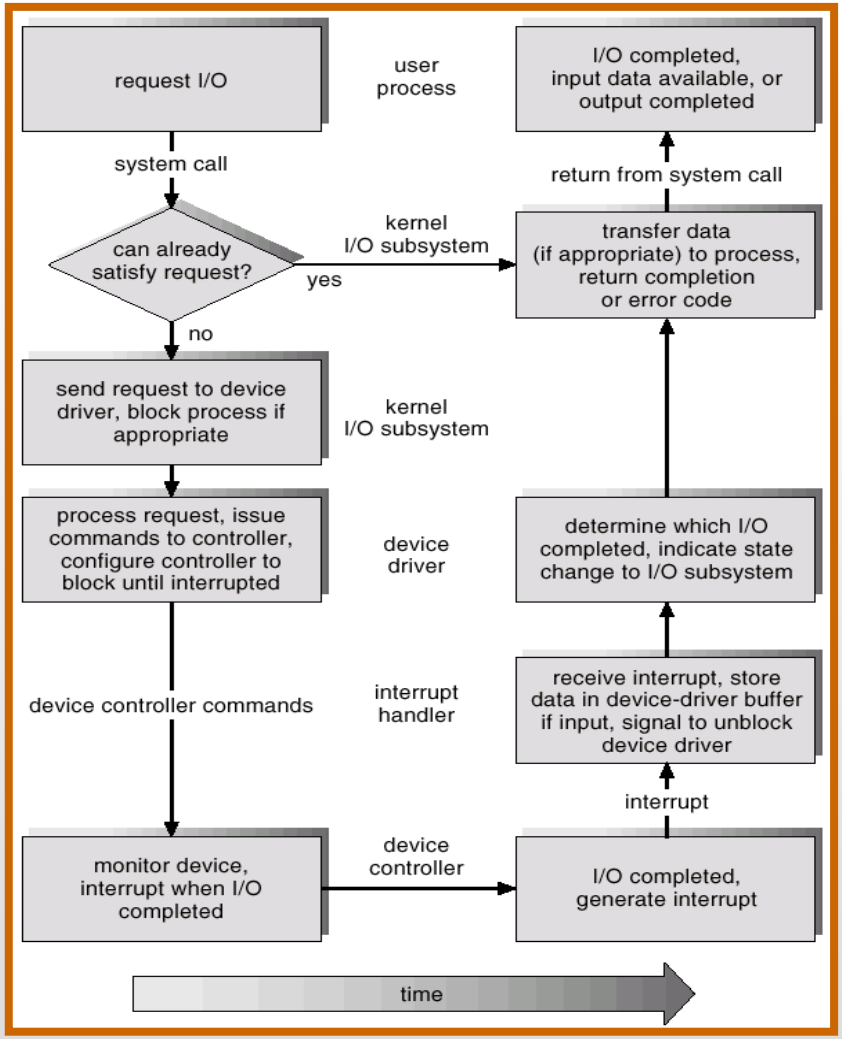
THE I/O SUBSYSTEM

block device switch table			
entry	open	close	strategy
0	gdopen	gdclose	gdstrategy
1	gtopen	gtclose	gtstrategy

character device switch table					
entry	open	close	read	write	ioctl
0	conopen	conclose	conread	conwrite	conioctl
1	dzbopen	dzbclose	dzbread	dzbwrite	dzbioctl
2	syopen	nulldev	syread	sywrite	syioctl
3	nulldev	nulldev	mmread	mmwrite	nodev
4	gdopen	gdclose	gdread	gdwrite	nodev
5	gtopen	gtclose	gtread	gtwrite	nodev

Figure 10.2. Sample Block and Character Device Switch Tables

Ciclo de vida de un requerimiento de I/O



Subsistema de I/O - Drivers

- Contienen el código dependiente del dispositivo
- Manejan un tipo dispositivo
- Traducen los requerimientos abstractos en los comandos para el dispositivo
  - Escribe sobre los registros del controlador
  - Acceso a la memoria mapeada
  - Encola requerimientos
- Comúnmente las interrupciones de los dispositivos están asociadas a una función del driver

Subsistema de I/O - Drivers

- Interfaz entre el SO y el HARD
- Forman parte del espacio de memoria del Kernel
  - En general 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

Driver - Ejemplo en Linux

- Linux distingue 3 tipos de dispositivos
  - Carácter: I/O programa 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.

Driver - Ejemplo en Linux (cont.)

- 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

Driver - Ejemplo en Linux (cont.)

- 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
  - .....

Driver - Ejemplo en Linux (cont.)

- 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)
```

Driver - Ejemplo en Linux (cont.)

- Acceso al hardware
  - Funciones para acceso a los puertos de I/O <asm/io.h>

```
unsigned char inb (unsigned short int port)
void outb (unsigned char value, unsigned short int port)
```

- Leen o Escriben un byte en el puerto de E/S indicado

```
En MS-DOS
---------------------
o 70 02
i 71
<retorna los minutos>

o 70 00
i 71
<retorna los segundos>
```

Performance

- I/O es uno de los factores que mas afectan a la performance del sistema:
  - Utiliza mucho la CPU para executar los drivers y el codigo del subsistema de I/O
  - Provoca Context switches ante las interrupciones y bloqueos de los procesos
  - Utiliza el bus de mem. en copia de datos:
    - Aplicaciones (espacio usuario) – Kernel
    - Kernel (memoria fisica) - Controladora

Mejorar la Performance

- ☑ Reducir el número de context switches
- ☑ Reducir la cantidad de copias de los datos mientras se pasan del dispositivo a la aplicación
- ☑ Reducir la frecuencia de las interrupciones, utilizando:
  - Transferencias de gran cantidad de datos
  - Controladoras mas inteligentes
  - Polling, si se minimiza la espera activa.
- ☑ Utilizar DMA