Sistemas Operativos

Dr. Luis Gerardo de la Fraga

E-mail: fraga@cs.cinvestav.mx
http://cs.cinvestav.mx/~fraga

Departamento de Computación
Cinvestav

12 de junio de 2015

Entrada y salida

- Una de las tareas principales de las computadoras es controlar los dispositivos de E/S.
- Se deben enviar comandos a los dispositivos, atrapar las interrupciones y manejar los errores.
- Y el SO debe proveer una interfaz entre los dispositivos y el resto del sistema que sea simple y fácil de usar.

Dispositivos de entrada y salida

- Se pueden dividir en dos categorías:
- Dispositivos de bloque
- Dispositivos de caracter

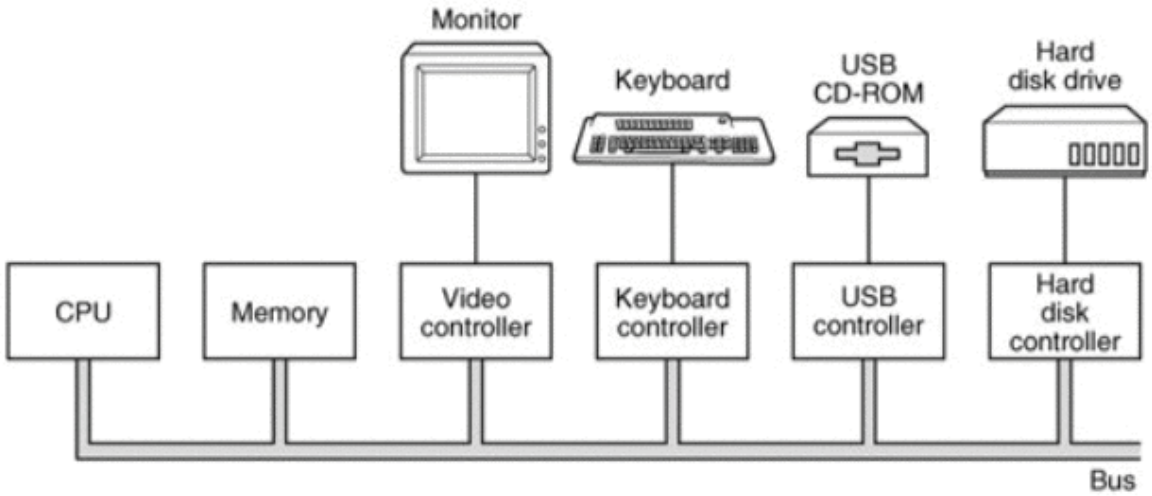
- Un dispositivo de bloque almacena la información en bloques de tamaño fijo, cada uno con su propia dirección.
- Los bloques pueden ser de tamaño de 512 a 32,768 bytes.
- Su propiedad principal es que se puede leer y escribir cada bloque independientemente de todos los demás.
- Los discos son los dispositivos de bloque más comunes

- Los dispositivos de caracter aceptan un flujo de caracteres, sin ninguna estructura de bloque.
- Dispositivos de caracter comunes son:
- El ratón,
- impresoras, e
- interfaces de red.

- Un reloj no pertenece a ninguna de estas dos categorías
- No son direccionables por bloque
- No generan o aceptan flujos de caracteres
- Producen interrupciones a intervalos bien definidos

Dispositivos	Velocidad de datos
Teclado	10 bytes/seg
Ratón	100 bytes/seg
Módem 56K	7 KB/seg
Escaner	400 KB/seg
Grabadora digital	4 MB/seg
CDROM 52x	8 MB/seg
FireWire (IEEE 1394)	50 MB/seg
USB 2.0	60 MB/seg
Monitor XGA	60 MB/seg
Red SONET OC-12	78 MB/seg
Ethernet Gigabit	125 MB/seg
Disco ATA serial	200 MB/seg
Disco SCSI Ultrawide 4	320 MB/seg
Bus PCI	528 MB/seg

- Los dispositivos de E/S generalmente se dividen en su parte mecánica y su parte electrónica
- La parte electrónica se le llama **controlador de dispositivo** o **adaptador**
- El sistema operativo trata con el controlador.



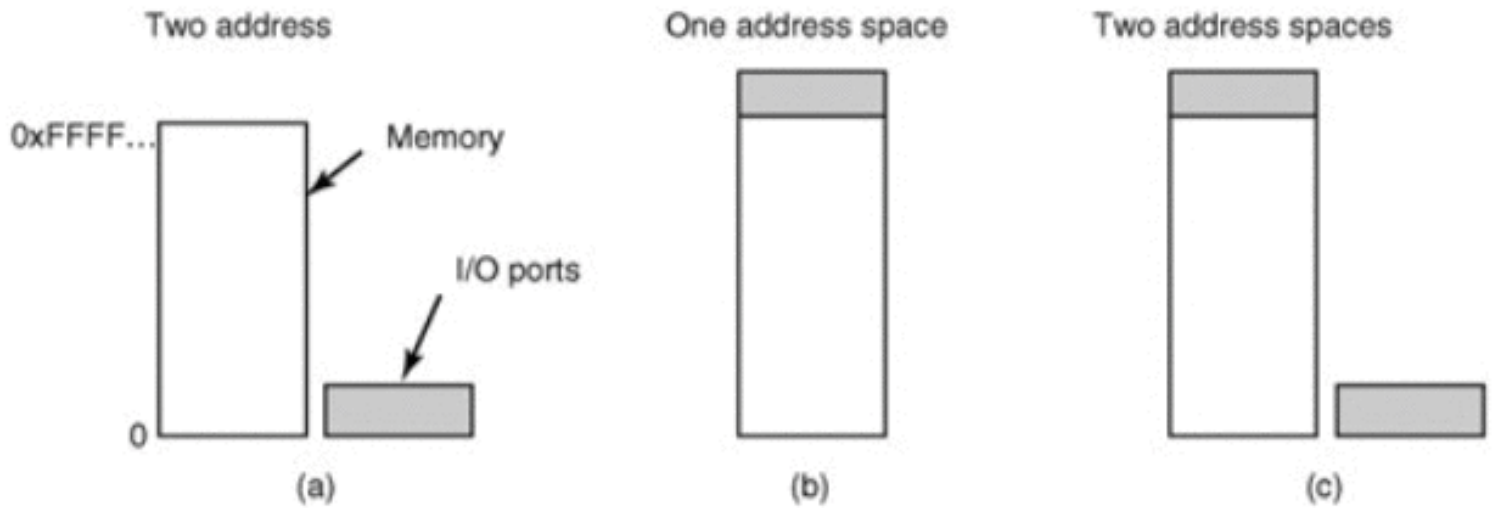
- La interfaz entre el controlador y el dispositivo generalmente es de bajo nivel.
- Para un disco, por ejemplo, podría estar formateado con 1024 sectores de 512 bytes por pista.
- Lo que sale del manejador, sin embargo, es un flujo serial de bits, comenzando con un preámbulo, y después los 4096 bits de un sector y finalmente la suma de chequeo.
- El preámbulo se escribe en el disco cuando éste se formatea y contiene el número de cilindro y de sector
- El trabajo del controlador es convertir el flujo serial de bits en un bloque de bytes y realizar la corrección de errores si es necesario.
- El bloque de bytes se ensambla por el controlador en un buffer interno.
- Si el bloque no contiene errores, entonces puede copiarse a memoria.

- Cada controlador tiene unos pocos registros que se usan para comunicarse con el CPU.
- Si se escribe a estos registros, el SO puede comandar al dispositivo para entregar o recibir datos, apagarse o encenderse o realizar alguna otra acción.
- Leyendo esos registros el SO puede saber cual es el estado del dispositivo, cuando está preparado para recibir comandos, etc.
- Además el controlador puede contener un buffer

¿Cómo se comunica el CPU con los registro de control y los buffers de datos?

- Existen dos alternativas:
- En una a cada registro de control se le asigna un número de puerto de E/S, un entero de 8 o 16 bits.
- Se usa la instrucción
  IN REG,PORT
  para que el CPU pueda leer el registro de control PORT y guardar el resultado en el registro REG
- De forma similar se usa la instrucción
  OUT PORT,REG

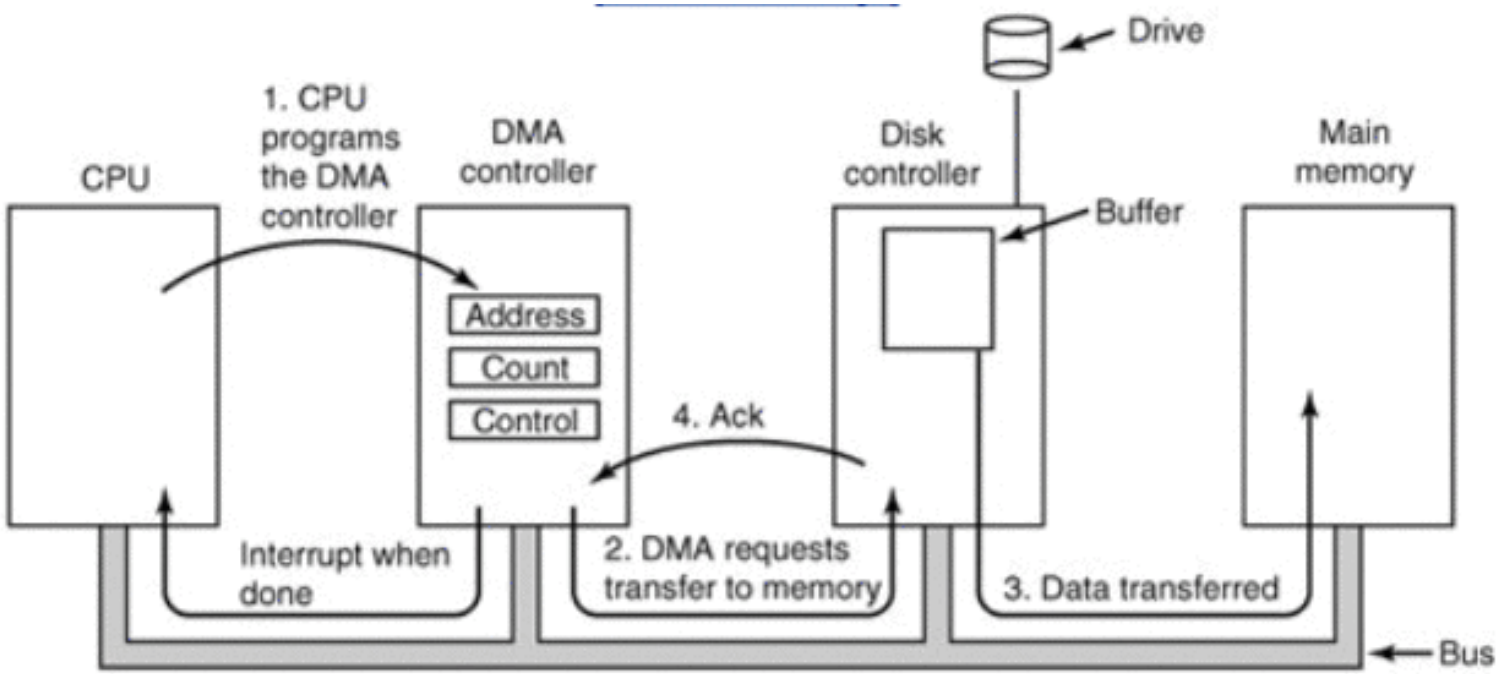
E/S mapeado en memoria



Interrupciones

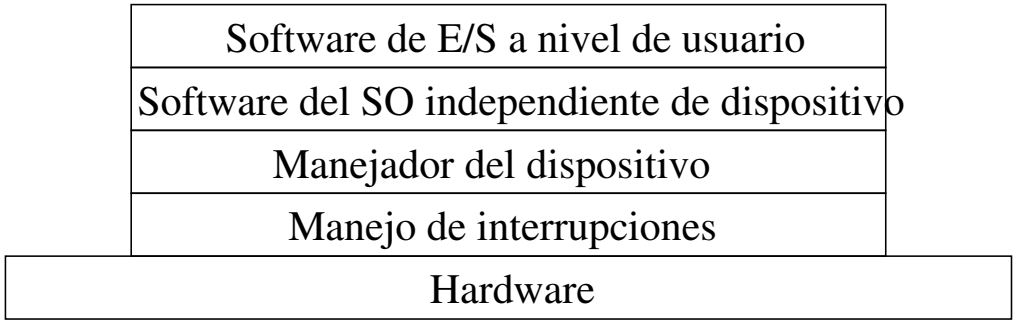
- Usualmente los registros del controlador tienen uno o más bit de estado que pueden probarse para determinar si una operación de salida está completa o si un nuevo dato del dispositivo de entrada está disponible
- El CPU puede realizar un ciclo, probar el estado del bit cada vez hasta que el dispositivo está listo para aceptar o proveer nuevos datos. A esto se la llama **espera ocupada** o *polling*.
- Además del bit de estado, mucho controladores usan interrupciones para decirle al CPU cuando están listos para usar sus registros para lectura o escritura
- El número de entradas del controlador de interrupciones puede estar limitado: las PC del tipo Pentium tienen solo 15 entradas disponibles para dispositivos de E/S

Acceso directo a memoria (DMA)



Principios del software para E/S

- Metas
- Manejador de interrupciones
- Manejadores de dispositivos
- Software de E/S independiente del dispositivo
- Software de E/S en el espacio de usuario



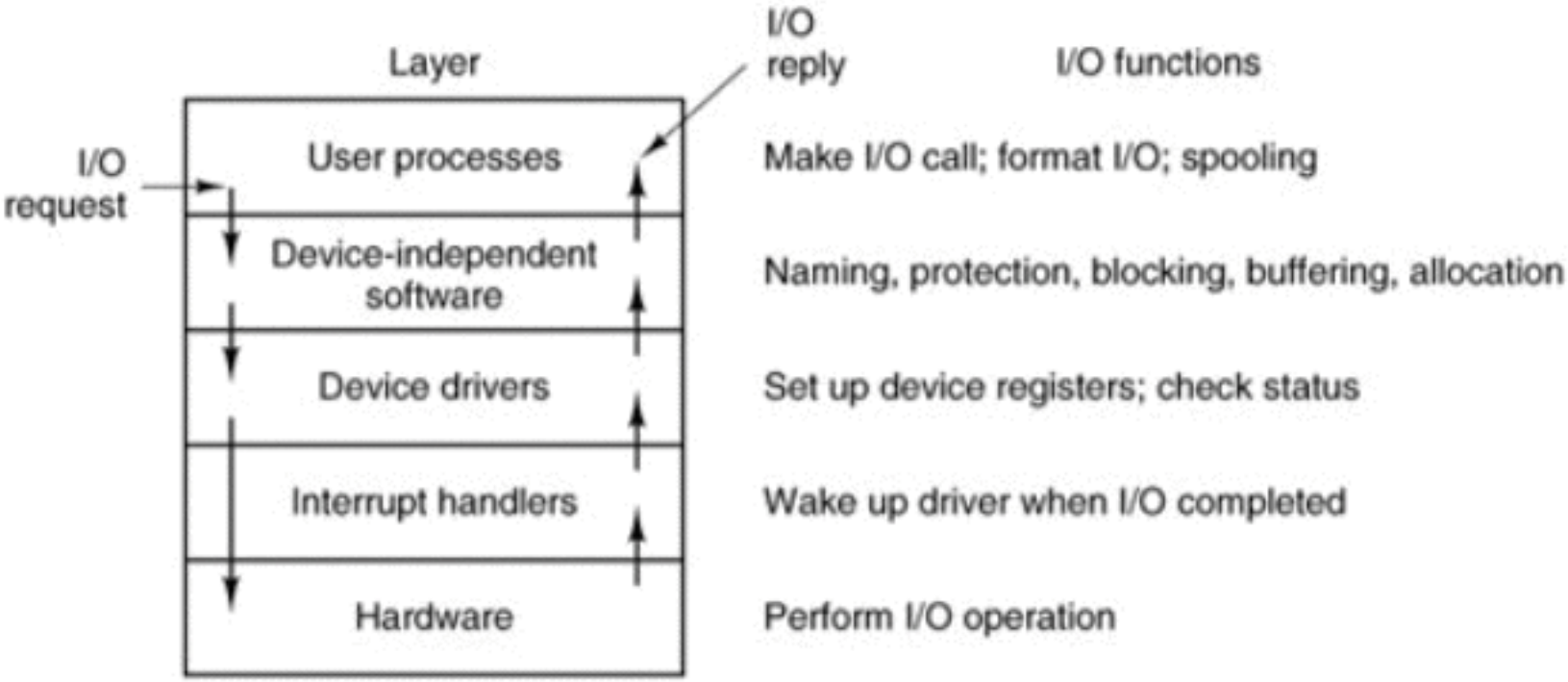
Capas de un sistema de software de E/S

Metas

- Independiente del dispositivo
- Nombramiento uniforme
- Manejo de errores
- Transferencias síncronas (bloqueantes) o asíncronas (no bloqueantes)
- Uso de buffers

Manejadores de dispositivos

- Cada dispositivo conectado al sistema necesita algún código específico para controlarlo.
- A este código se la llama **manejador de dispositivo**, en general es escrito por el vendedor del dispositivo y entregado con el mismo dispositivo.
- Tradicionalmente el manejador de dispositivos ha sido parte del núcleo del sistema.
- Esto da el mejor rendimiento y la peor confiabilidad.
- Un fallo en un manejador debido a un bug puede tirar el sistema completo.


Layer
I/O reply
I/O functions
I/O request
User processes
Make I/O call; format I/O; spooling
Device-independent software
Naming, protection, blocking, buffering, allocation
Device drivers
Set up device registers; check status
Interrupt handlers
Wake up driver when I/O completed
Hardware
Perform I/O operation

<table>
<tr><td rowspan="4">User mode</td><td>User applications</td><td colspan="5">For example, bash, LibreOffice, Apache OpenOffice, Blender, 0 A.D., Mozilla Firefox, etc.</td></tr>
<tr><td>Low-level system components:</td><td>System daemons: systemd, runit, logind, networkd, soundd...</td><td>Windowing system: X11, Wayland, Mir, SurfaceFlinger (Android)</td><td colspan="2">Other libraries: GTK+, Qt, EFL, SDL, SFML, FLTK, GNUstep, etc.</td><td>Graphics: Mesa 3D, AMD Catalyst, ...</td></tr>
<tr><td>C standard library</td><td colspan="5"><code>open()</code>, <code>exec()</code>, <code>sbrk()</code>, <code>socket()</code>, <code>fopen()</code>, <code>calloc()</code>, ... (up to 2000 subroutines)
glibc aims to be POSIX/SUS-compatible, uClibc targets embedded systems, bionic written for Android, etc.</td></tr>
<tr><td rowspan="3">Kernel mode</td><td rowspan="3">Linux kernel</td><td colspan="5"><code>stat</code>, <code>splice</code>, <code>dup</code>, <code>read</code>, <code>open</code>, <code>ioctl</code>, <code>write</code>, <code>mmap</code>, <code>close</code>, <code>exit</code>, etc. (about 380 system calls)
The Linux kernel System Call Interface (SCI, aims to be POSIX/SUS-compatible)</td></tr>
<tr><td>Process scheduling subsystem</td><td>IPC subsystem</td><td>Memory management subsystem</td><td>Virtual files subsystem</td><td>Network subsystem</td></tr>
<tr><td colspan="5">Other components: ALSA, DRI, evdev, LVM, device mapper, Linux Network Scheduler, Netfilter
Linux Security Modules: SELinux, TOMOYO, AppArmor, Smack</td></tr>
<tr><td colspan="7">Hardware (CPU, main memory, data storage devices, etc.)</td></tr>
</table>