

Administración de E/S

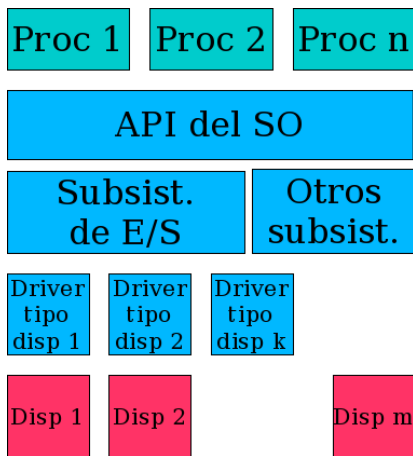
Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

Sistemas Operativos, primer cuatrimestre de 2016

(2) Esquema de E/S

- Para nosotros, un *dispositivo* de E/S va a tener, conceptualmente, dos partes:
 - El dispositivo *físico*.
 - Un *controlador del dispositivo*: interactúa con el SO mediante algún tipo de bus o registro.

(3) Arquitectura del subsistema de E/S



(4) Los drivers

- Los drivers son componentes de software muy específicos.
- Conocen las particularidades del dispositivo de HW contra el que hablan.
- Incluso distintos modelos de un mismo fabricante pueden requerir distintos drivers.
- Ejemplo: ¿Para indicar fin de la operación hay que leer el segundo o el cuarto bit? Esa información sólo la conoce el driver.
- Los drivers son clave.
 - Corren con máximo privilegio: pueden hacer colgarse a todo el sistema.
 - De ellos depende el rendimiento de E/S, que es fundamental para el rendimiento combinado del sistema.

(5) Interacción con los dispositivos

- Polling
 - El driver periódicamente verifica si el dispositivo se comunicó.
 - Ventajas: sencillo, cambios de contexto controlados.
 - Desventajas: Consume CPU.
- Interrupciones (o push)
 - El dispositivo avisa (genera una interrupción).
 - Ventajas: eventos asincrónicos poco frecuentes.
 - Desventajas: cambios de contexto impredecibles.
- DMA (acceso directo a memoria)
 - Para transferir grandes volúmenes (la CPU no interviene).
 - Requiere de un componente de HW, el controlador de DMA.
 - Cuando el controlador de DMA finaliza, interrumpe a la CPU.

(6) Grupos de dispositivos

Char device

Dispositivos en los cuales se transmite la información byte a byte. Ejemplos: mouse, teclado, terminales o puerto serie. Debido a su acceso secuencial (byte a byte) no soportan acceso aleatorio y no utilizan *cache*.


Block device

Dispositivos en los cuales se transmite la información en bloque. Ejemplos: disco rígido, flash memory o CD-ROM. Permite el acceso aleatorio y por lo general utilizan un buffer (*cache*).

(7) Linux /dev

crw-rw-rw-	1	root	wheel	17, 1	Apr 27 07:41	cu.Bluetooth
b rw-r---	1	root	operator	1, 0	Apr 27 07:41	disk0
crw-rw-rw-	1	root	wheel	24, 2	Apr 27 07:41	dtrace
c rw-r---	1	root	operator	1, 0	Apr 27 07:41	rdisk0


(8) Subsistema de E/S

- El diálogo con estos dispositivos tiene las siguientes características:
 - Son de lectura, escritura o lecto-escritura.
 - Brindan acceso secuencial o aleatorio (sería mejor decir arbitrario).
 - Son compartidos o dedicados.
 - Permiten una comunicación de a caracteres o de a bloques.
 - La comunicación con ellos es sincrónica o asincrónica.
 - Tienen distinta velocidad de respuesta.
- Una de las funciones del SO, en tanto API de programación, es brindar un acceso consistente a toda la fauna de dispositivos ocultando las particularidades de cada uno de ellos tanto como sea posible. 

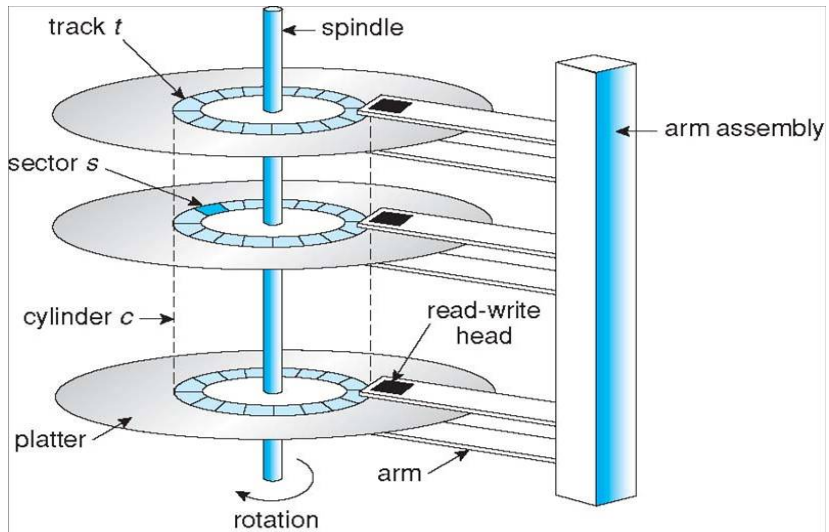
(9) Subsistema de E/S

- Se ocupa de proveerle al programador una API sencilla:
 - `open()` / `close()`
 - `read()` / `write()`
 - `seek()`
- Sin embargo, hay cosas que no se pueden (o deben) ocultar.
Ejemplo: algunas aplicaciones necesitan enterarse si no lograron acceso exclusivo a un dispositivo.
- La misión del SO es hacer esto de manera correcta y eficiente.
- Esa responsabilidad está compartida entre el *manejador de E/S* y los *drivers*. ⚠


(10) Subsistema de E/S

- Todo es un archivo 
- Se proveen funciones de *alto* nivel para acceso a archivos:
 - `fopen`, `fclose`
 - `fread`, `fwrite`: Leer/Escribir archivos en modo bloque.
 - `fgetc`, `fputc`: Leer/Escribir archivos en modo char.
 - `fgets`, `fputs`: Leer/Escribir archivos en modo char stream.

(11) Planificación de E/S a disco



(12) Planificación de E/S a disco

- Manejar la cola de pedidos de E/S para lograr el mejor rendimiento posible. 
- Maximizar el *ancho de banda*:
cantidad de bytes que se pueden transferir a la vez.
- Minimizar el *tiempo de búsqueda* (*seek time*):
tiempo para que la cabeza se ubique sobre el cilindro que tiene el sector buscado.

(13) Políticas de scheduling de E/S a disco

- *FIFO* o *FCFS* (*First Come, First Served*).
 - Ventajas: simple.
 - Desventajas: tiempo de búsqueda incontrolable.
- *SSTF*: *Shortest Seek Time First*.
 - Algoritmo goloso: atender el próximo pedido más cercano a donde está la cabeza en ese momento.
 - Ventajas: mejora los tiempos de respuesta (pero no es óptimo).
 - Desventaja: puede producir inanición.
- *Scanning* o *del ascensor* (*elevator*).
 - Idea: ir primero en un sentido, atendiendo los pedidos que encuentro en el camino, luego en el otro.
 - Ventajas: reduce los movimientos.
 - Desventajas: el tiempo de espera no es tan uniforme.
- En la práctica, ninguno se utiliza de manera pura.
- Hay prioridades: bajar páginas de cachés, swapping...

(14) Otras estrategias: Spooling

- Simultaneous Peripheral Operation On-Line
- *Spooling* es una forma de manejar a los dispositivos que requieren acceso dedicado en sistemas multiprogramados. ⚠
- El caso típico es la impresora.
- El kernel no se entera de que se está haciendo spooling. El usuario sí. ⚠
- Line Printer Daemon Protocol
<https://tools.ietf.org/html/rfc1179>
- Design Goals for an Internet Printing Protocol
<https://tools.ietf.org/html/rfc2567>

(15) Otros usos de E/S: Locking


- POSIX garantiza que `open(..., O_CREAT | O_EXCL)` es atómico y crea el archivo si no existe o falla si ya existe.
- Eso brinda un mecanismo sencillo, aunque no extremadamente eficiente, de exclusión mutua.
- Suele ser usado para implementar locks.

Redundancia

Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

Sistemas Operativos, primer cuatrimestre de 2016

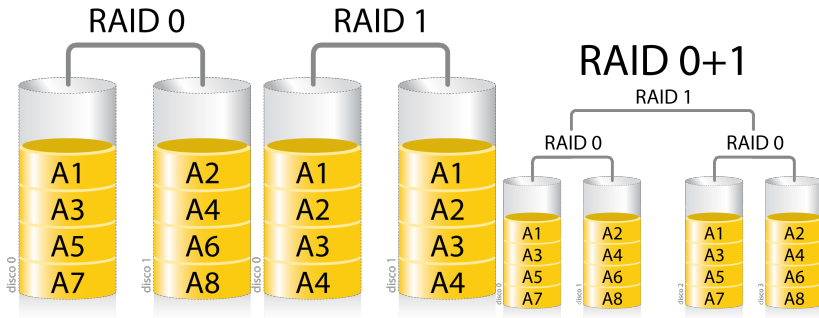
(17) Redundancia

- Un método muy común es *RAID: Redundant Array of Inexpensive Disks*. 
- La idea, en su forma más elemental es usar dos discos: cada escritura se hace en los dos. Si uno se rompe, tengo el otro.
- Esta alternativa se llama *espejo* o *mirror* y si bien es conveniente, puede ser muy costosa.
- Sin embargo, aporta una ventaja adicional. Puedo hacer dos lecturas a la vez, una en cada disco.
- En realidad, hay varios niveles de RAID, que tienen diferentes ventajas/desventajas en cuanto a rendimiento y redundancia.

(18) Niveles de RAID

- RAID 0 (*stripping*):
 - No aporta redundancia.
 - Pero mejora el rendimiento: los bloques de un mismo archivo se distribuyen en dos (o más) discos.
 - Mejora el ancho de banda, permite escrituras en paralelo (si los discos están en diferentes controladoras).
- RAID 1 (*mirroring*):
 - Espejado de los discos.
 - Mejora el rendimiento de las lecturas.
 - Escrituras: mejor caso lo mismo, peor caso el doble.
 - Es muy caro.
- RAID 0+1:
 - Combina los dos anteriores: espejado y stripping.
 - Archivo espejado. Lecturas paralelas en cada disco.
 - Ventaja: Lecturas más rápidas, como en stripping.
 - Desventaja: Escrituras más lentas, como en mirroring.

(19) Niveles de RAID



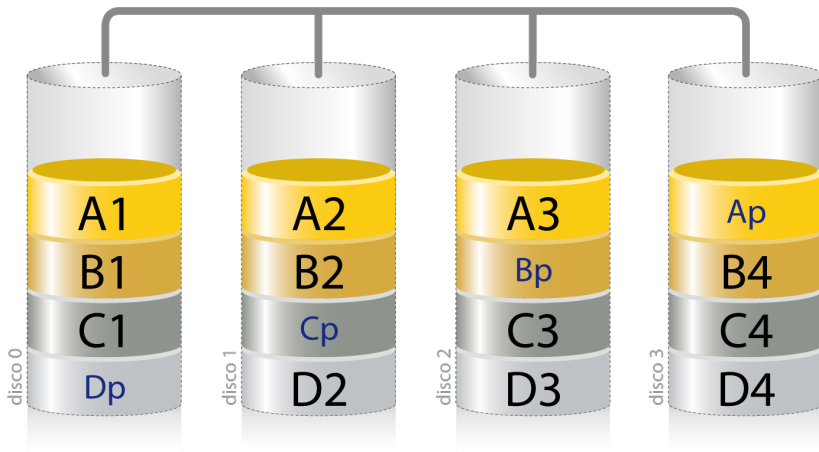
(20) Niveles de RAID (cont.)

- RAID 2 y 3:
 - La idea es tener, por cada bloque, guardada información adicional que permita determinar si se dañó o no.
 - Además, cierto tipo de errores se pueden corregir automáticamente, recomutando el bloque dañado a partir de la información redundante.
 - Adicionalmente, cada bloque lógico se distribuye entre todos los discos participantes.
 - RAID 2 requiere 3 discos de paridad por cada 4 de datos mientras que RAID 3 requiere sólo 1.
 - Sin embargo, todos los discos participan de todas las E/S, lo cual lo hace más lento que RAID 1.
 - Puede requerir mucho procesamiento para computar las redundancias.
 - Por eso se suele implementar por HW en una controladora dedicada, al igual que todos los niveles siguientes.

(21) Niveles de RAID (cont.)

- RAID 4:
 - Es cómo RAID 3, excepto que hace el stripping a nivel de bloque (ie, cada bloque en un solo disco).
 - El disco dedicado a paridad sigue siendo un cuello de botella para el rendimiento, porque todas las escrituras lo necesitan.
- RAID 5:
 - Junto con 0, 1, y 0+1 es de los más usados en la práctica.
 - Usa datos redundantes, pero los distribuye en $N+1$ discos.
 - Es decir, no hay un disco que sólo contenga redundancia.
 - Cada bloque de cada archivo va a un disco distinto.
 - Para cada bloque, uno de los discos tiene los datos y otro tiene la información de paridad.
 - Si bien ya no hay cuello de botellas para las escrituras hay que mantener la paridad distribuida, lo que no es sencillo.
 - Puede soportar la pérdida de un disco cualquiera.
 - Cuando se reemplaza y comienza la reconstrucción, el rendimiento se degrada notablemente.

RAID 5



- RAID 6

- Es como RAID 5, pero agrega un segundo bloque de paridad, también distribuido entre todos los discos.
- Las implementaciones varían, pero el objetivo principal es soportar la rotura de hasta dos discos.
- Considerando que RAID 5 se suele usar con un *hot spare*, no hay diferencia sustancial en el espacio “desperdiciado” (a grandes rasgos).