

# Administración de E/S

Diego Fernandez Slezak

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

Sistemas Operativos, primer cuatrimestre de 2024

## (2) Hasta ahora...

- Hablamos varias veces sobre hacer E/S.
- Los dispositivos de E/S pueden ser categorizados como de almacenamiento, comunicaciones, interface de usuario y otros.
- Nos vamos a concentrar principalmente en los dispositivos de almacenamiento.

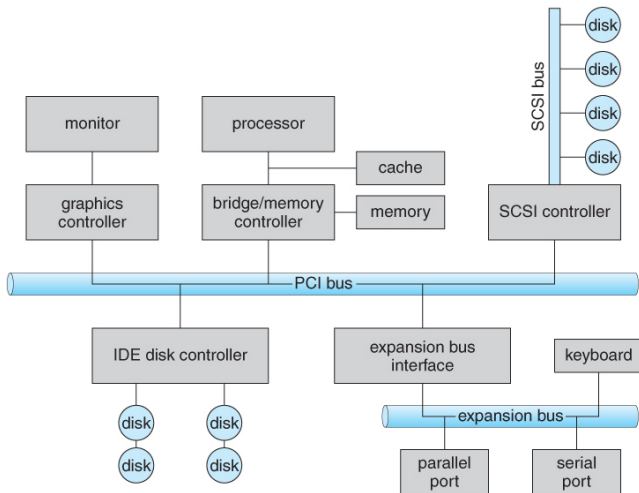
### (3) Con qué lidiamos

- Tradicionalmente, con respecto al almacenamiento, los SO se tenían que preocupar de:
  - Discos rígidos. Siguen siendo la preocupación primaria.
  - Unidades de cinta. Actualmente se usan más que nada para hacer copias de seguridad.
  - Discos removibles: disquettes, CDs, DVDs.
- También hay discos virtuales, que están en otro punto de la red y a los que se accede mediante ella.
  - NFS, CIFS, DFS, AFS, Coda por nombrar algunos.
  - De manera genérica se los llama *NAS: Network Attached Storage*.
- Existen también otras alternativas :
  - *Storage Area Network (SAN)*: se trata de tener el almacenamiento en la red, pero una red especial, donde los protocolos son específicos para este tipo de datos (son de más bajo nivel).

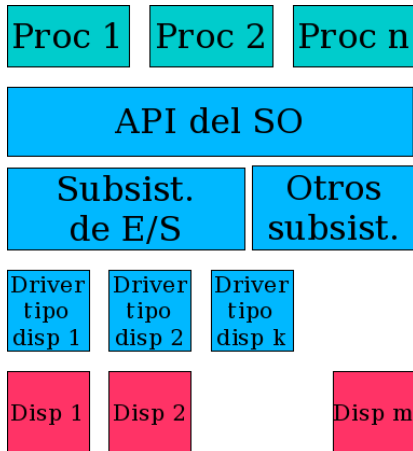
## (4) Esquema de E/S

- 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.

## (5) Estructura típica de BUS de PC



## (6) Administrador de E/S



## (7) Los drivers


- Los drivers son componentes de software muy específicos.
- Conocen las particularidades del 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.

## (8) 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.



## (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

- Los dispositivos pueden separarse en dos grupos:
  - Char device
  - Block device

### 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*).


## (11) 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


## (12) 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. 

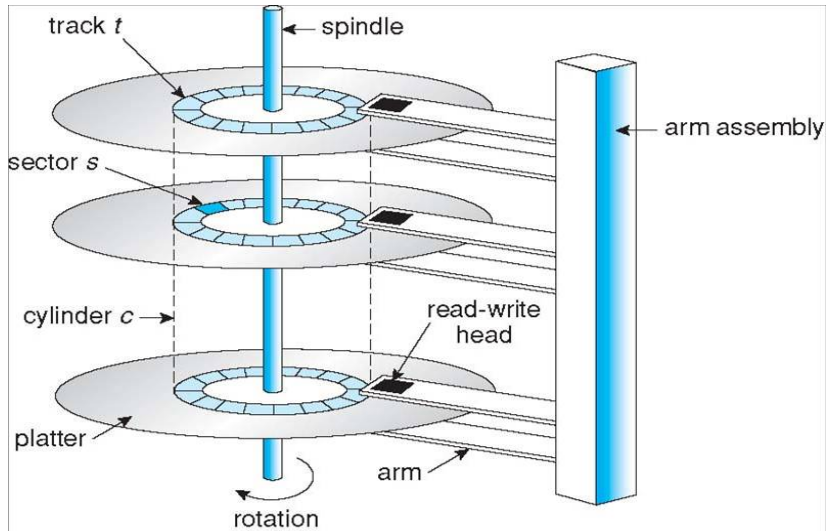
## (13) API del Subsistema de E/S

- 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.
  - `fscanf`, `fprintf`: Leer/escribir archivos en modo char con formato.

## (14) Planificación de E/S

- Una de las claves para obtener un buen rendimiento de E/S es manejar apropiadamente el disco.
- Recordemos: hay una cabeza que se mueve. Moverla toma tiempo.
- Queremos minimizar esos movimientos.
- Pero los pedidos de E/S a disco llegan constantemente, incluso antes de que terminemos con uno.
- Por eso, la planificación de disco se trata de cómo manejar la cola de pedidos de E/S para lograr el mejor rendimiento posible. 
- Es decir, no sólo hay que manejar el *ancho de banda*, que es la cantidad de bytes que se pueden transferir a la vez, y la *latencia rotacional*, que es el tiempo necesario para que el disco rote y la cabeza quede sobre el sector deseado.
- Lo más importante es el *tiempo de búsqueda* o *seek time*, que es el tiempo necesario para que la cabeza se ubique sobre el cilindro que tiene el sector buscado.

## (15) Mecanismo de movimiento de cabezas de disco



## (16) Políticas de scheduling de E/S a disco

- El esquema más simple es *FIFO* o *FCFS* (*First Come, First Served*).
- Imaginemos que la cola tiene pedidos para los cilindros: 20, 200, 10.
- Problema, la cabeza va de acá para allá, como bola sin manija.
- Otro esquema posible es *SSTF*: *Shortest Seek Time First*.
- La idea es atender como próximo pedido al más cercano a donde está la cabeza en el momento.
- Si bien mejora los tiempos de respuesta, puede producir inanición.
- Notar que es un algoritmo goloso, pero no es óptimo.



## (17) Políticas de scheduling de E/S a disco

- Otra posibilidad es el algoritmo *scan* o *del ascensor* (*elevator*).  
Idea: ir primero en un sentido, atendiendo los pedidos que encuentro en el camino, luego en el otro.
- Podría suceder que llegue una solicitud para el cilindro inmediato anterior, pero tenga que esperar a que cambie de dirección.
- Además, el tiempo de espera no es tan uniforme.
- En la práctica, ninguno de estos algoritmos se utiliza de manera pura: hay prioridades (por ejemplo, bajar páginas de cachés, o swapping de procesos), etc.

## (18) Primer HD de IBM, circa 1956

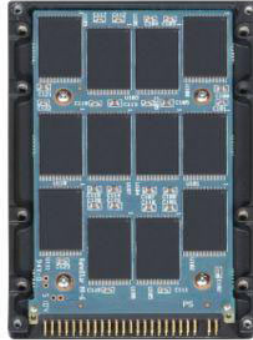


5M (7 bit) caracteres - Tiempo de Acceso =  $< 1$  segundo

## (19) La tecnología evoluciona



Traditional hard disk drive



Solid state hard drive

## (20) SSD - Solid State Drive

- Los “discos” de estado sólido han mejorado sus prestaciones y disminuido su precio.
- Son más livianos, resistentes y silenciosos. Consumen menos energía.
- Tienen mejor performance en la lectura que los HDD, al no tener componentes mecánicos. La escritura es más compleja.
- Problema de durabilidad y write amplification.

## (21) Gestión del disco

- Formateo:

- Se trata de poner en cada sector unos códigos que luego sirven a la controladora de disco para efectuar detección y corrección de errores.
- Funcionan como un prefijo y un postfijo a la parte donde efectivamente van los datos en cada sector.
- Si al leer un sector, el prefijo y postfijo no tienen el valor que deberían, el sector está dañado.

- Booteo:

- Las computadoras suelen tener un minúsculo programa en ROM que carga a memoria ciertos sectores del comienzo del disco, y los comienza a ejecutar.
- El programa cargado es muy pequeño. No llega a ser el SO, sino más bien un cargador del SO.

## (22) Gestión del disco (cont.)

- Bloques dañados:
  - A veces se manejan por software, y el sistema de archivo es responsable de anotar los inválidos. Ejemplo: FAT.
  - Los discos SCSI vienen con sectores extra para reemplazar a los defectuosos.
  - Cuando la controladora detecta un bloque dañado actualiza una tabla interna de remapeo y utiliza otro sector.
  - Para no interferir con las optimizaciones del scheduler de E/S, los discos a veces traen sectores extra en todos los cilindros.

## (23) Spooling


- *Spooling* es una forma de manejar a los dispositivos que requieren acceso dedicado en sistemas multiprogramados. ⚠
- El caso típico es la impresora.
- Cuando un usuario manda a imprimir no queremos que se bloquee hasta que terminen de imprimir los demás.
- La idea es poner el trabajo en una cola, y designar un proceso que los desencole a medida que el dispositivo se libere.
- Notar:
  - El kernel no se entera de que se está haciendo spooling.
  - El usuario sí. ⚠
- El nombre viene de *Simultaneous Peripheral Operation On-Line*.

## (24) 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.



## (25) Protección de la información

- ¿Tiene sentido proteger la información? 
- La pregunta que debe responderse es cuál es el valor de la información que estoy protegiendo:
  - Cuánto vale para mí.
  - Qué pasa si se pierde.
  - Qué cosas no puedo hacer sin ella.
- En base a eso debo asignarle un valor a la información y tomar una política de resguardo.
- En general esta política va a tener un costo (económico y/o en términos de esfuerzo) proporcional al valor de la información.
- Algunas estrategias de protección:
  - MSSVR: “¡Mirá si se va romper!”
  - Estudios recientes demostraron que no suele ser muy efectiva.
  - La estrategia del medioevo: organizar una cadena de oración en la oficina para que el disco no reviente.
  - Hay métodos más sofisticados...


## (26) Copias de seguridad

- Hacer una copia de seguridad (*backup*) consiste en resguardar todo lo importante en otro lado. 
- Cosas que hay que hacer todos los días sí o sí (en orden de importancia):
  - 1) *Hacer backup.*
  - 2) Respirar.
  - 3) Otras actividades de menor importancia.
- Se suele hacer en cinta, incluso en bibliotecas de cintas robotizadas.
- Toma tiempo, y por eso se suele programar a los sistemas para que lo hagan por la noche.
- Otra estrategia consiste en copiar los datos a otro disco. Si es removible mejor.

## (27) Copias de seguridad (cont.)

- Copiar *todos* los datos puede ser muy costoso.
- Una estrategia muy común consiste en:
  - Una vez al {mes|semana|etc.} hacer una copia *total*.
  - Todas las noches realizar una copia *incremental*: sólo los archivos modificados desde la última copia incremental.
  - Alternativamente, realizar una copia *diferencial*: sólo los archivos modificados desde la última copia total.
- Para restaurar:
  - Si hago sólo copias totales, tomo la del día correspondiente y listo.
  - Si hago copias diferenciales, necesito la última copia total más la última diferencial.
  - Si lo que tengo son incrementales, necesito la última copia total y todas las incrementales entre ésa copia total y la fecha requerida.
- Es decir:
  - Hoy = Último total + último diferencial
  - Hoy = Último total +  $\sum_i \text{Incremental}_i$

## (28) Redundancia

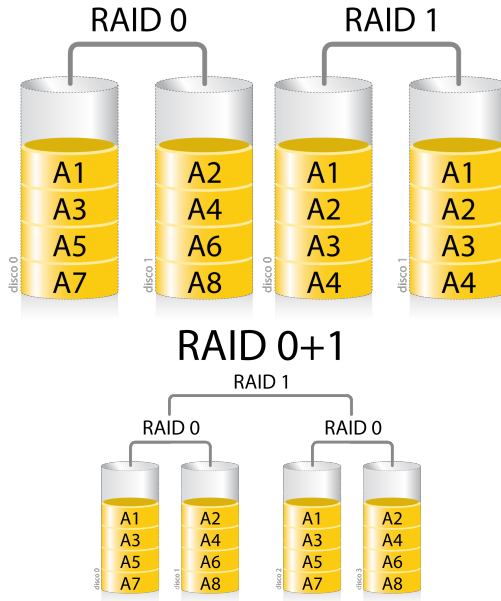
- A veces una copia de seguridad no alcanza.
- El costo de que el sistema salga de línea es muy alto.
- Entonces, conviene implementar redundancia.
- Un método muy común es *RAID: Redundant Array of Inexpensive Disks*. 
- La idea, en su forma más elemental es usar dos discos: cada escritura se hace en 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.

## (29) 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.
  - Las escrituras, en mejor caso, tardan lo mismo, en peor, el doble.
  - Es muy caro.
- RAID 0+1:
  - Combina los dos anteriores: espejado y stripping.
  - Es decir: cada archivo está espejado, pero al leerlo leo un bloque de cada disco.
  - Lo leo más rápido que en mirroring simple. Como si fuera stripping.

Para el espejado, tengo que escribir cada bloque en ambos

## (30) Niveles de RAID



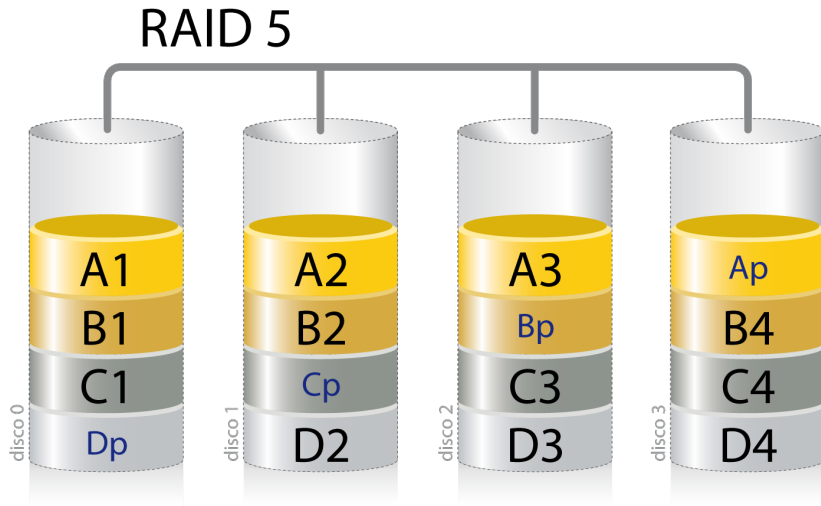
## (31) Niveles de RAID (cont.)

- 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, recomputando 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.

## (32) 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.
  - También 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 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).

## (35) Sin embargo...

- RAID no protege contra borrar (o modificar) un archivo accidentalmente.
- Por eso se combina con copias de seguridad (no son excluyentes). 
- Si la aplicación corrompe los datos, ningún mecanismo sirve.
- Si se corrompe la estructura interna de los archivos, RAID tampoco ayuda.
- Para eso hay sistemas de archivos que brindan algo de protección.