

CAPÍTULO 3 ENTRADA / SALIDA

3.1 Introducción

Una de las principales funciones de un SO es la de controlar todos los dispositivos de E/S de una computadora.

El código de E/S representa una parte significativa del SO total.

3.2 Principios del hardware de E/S

3.2.1 Dispositivos de E/S

Los dispositivos de E/S se pueden dividir a grandes rasgos en dos categorías, pero esta clasificación no es perfecta:

1. Dispositivos por bloques: estos almacena la información en bloques de tamaño fijo, cada uno con su propia dirección.
2. Dispositivos por caracteres: estos suministran o aceptan una corriente de caracteres, sin contemplar ninguna estructura de bloques, no es direccionable y no tiene una operación de búsqueda.

Algunos dispositivos no se ajustan a esta clasificación.

El software de bajo nivel que administra la parte dependiente del dispositivo se denomina **controlador**.

3.2.2 Controladores de dispositivos

El SO casi siempre trata con el controlador del dispositivo más que con el dispositivo mismo.

La mayor parte de las computadoras usan el modelo de bus único para la comunicación entre la CPU y los controladores.

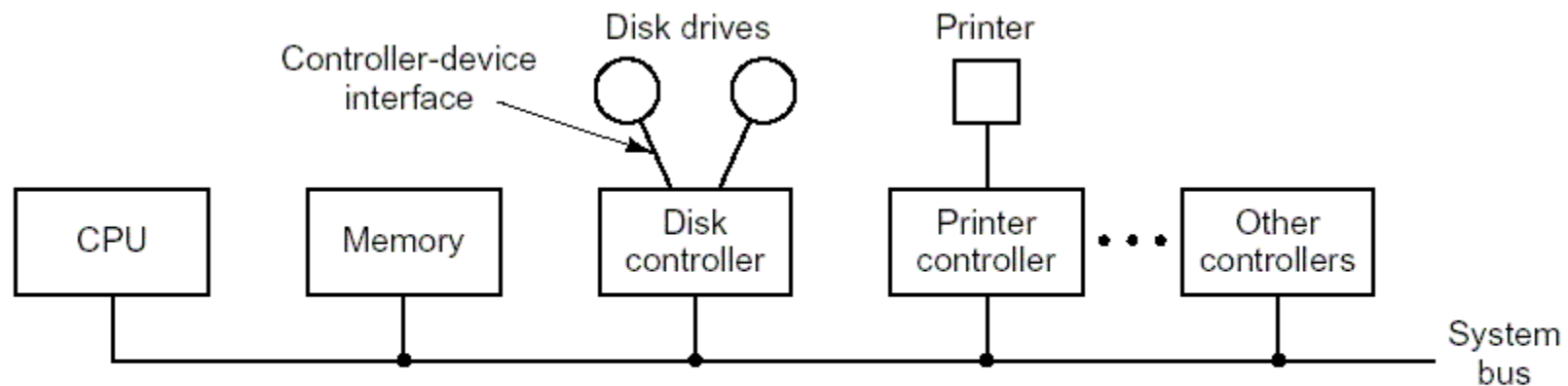


Figure 3-1. A model for connecting the CPU, memory, controllers, and I/O devices.

Las macro computadoras (*mainframes*) con frecuencia usan un modelo diferente con múltiples buses y canales de E/S.

Cada controlador tiene unos cuantos registros para comunicarse con la CPU. En algunas computadoras con **E/S mapeada en memoria**, estos registros forman parte del espacio de direcciones de la memoria normal.

Una interrupción (un suceso eléctrico), es un mecanismo que utilizan muchos controladores para indicarle a la CPU que están listos para proporcionar o recibir información.

3.2.3 Acceso directo a memoria (DMA)

Muchos controladores, sobre todo los de dispositivos por bloques, manejan el DMA, que es un mecanismo para liberar a la CPU de trabajo de bajo nivel.

El DMA es el dispositivo encargado de hacer las transferencias de memoria a disco y viceversa.

No todas las computadoras usan DMA, dado que el controlador DMA es mucho más lento que la CPU.

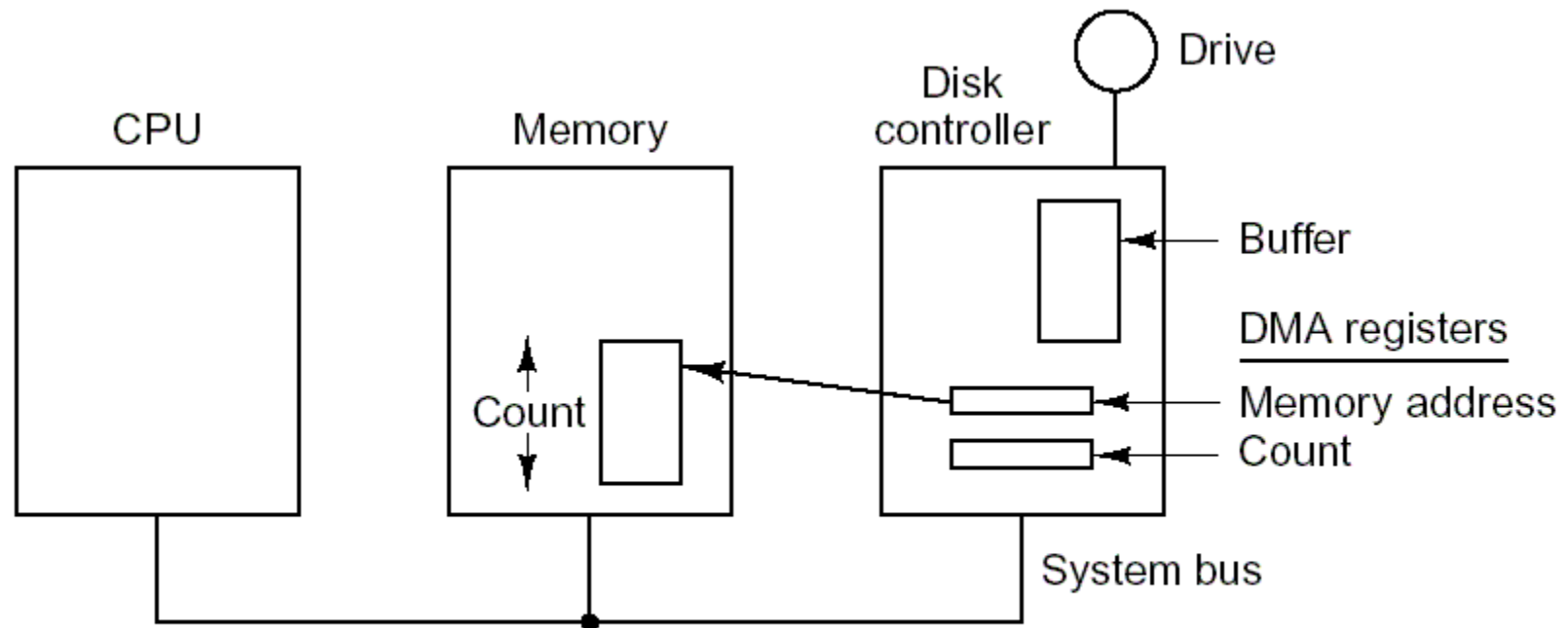


Figure 3-3. A DMA transfer is done entirely by the controller.

Si el tiempo que toma transferir un bloque del controlador a la memoria es más largo que el que toma leer un bloque de disco,

puede ser necesario leer un bloque y después saltarse dos o más bloques. A esto último se le llama **intercalación**.

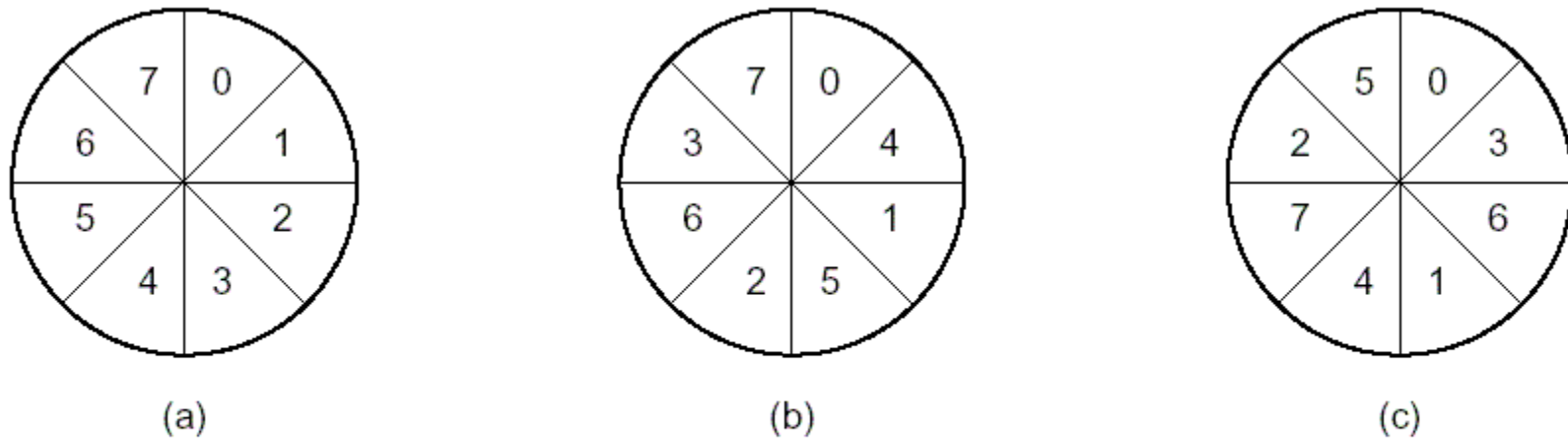


Figure 3-4. (a) No interleaving. (b) Single interleaving. (c) Double interleaving.

3.3 Principios de software de E/S

Los objetivos generales del SW de E/S son fáciles de plantear.

La idea básica es organizar el SW como una serie de capas con un nivel de abstracción cada vez mayor.

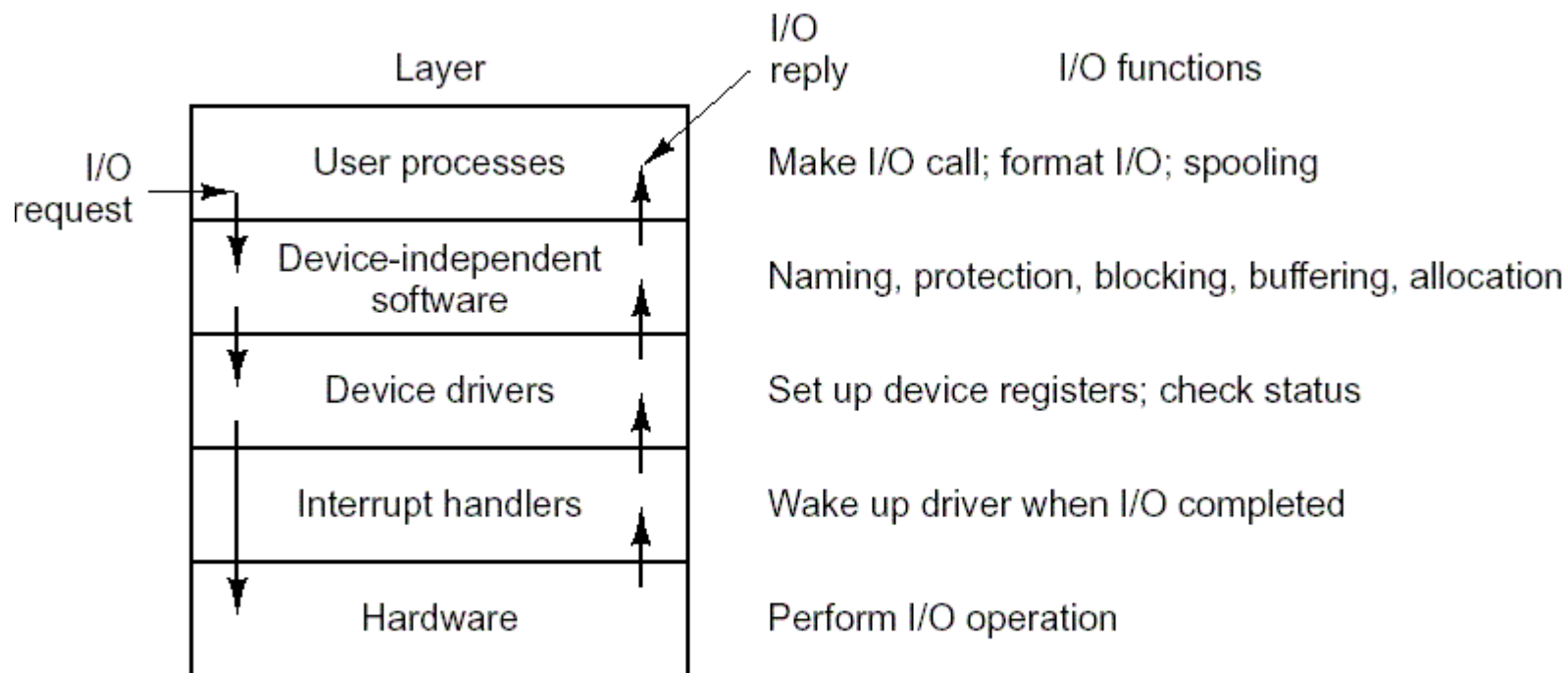


Figure 3-6. Layers of the I/O system and the main functions of each layer.

3.4 Bloqueo mutuo

Los sistemas de cómputo tienen muchos recursos que sólo pueden ser usados por un proceso a la vez. Por ello, todos los SO tienen la capacidad de conceder temporalmente a un proceso acceso exclusivo a ciertos recursos.

Un conjunto de procesos está en **bloqueo mutuo** si cada proceso del conjunto está esperando un evento que sólo otro proceso del conjunto puede causar.

3.4.1 Recursos.

Deberemos entender como recursos a los objetos otorgados, esto es, cualquier cosa que sólo puede ser usada por un proceso en un instante dado.

Los recursos son de dos tipos:

1. Expropiables: se le puede arrebatar al proceso que lo tiene sin que haya efectos adversos.
2. No expropiables: no puede quitársele a su poseedor actual sin provocar problemas como inconsistencia de información u otras consecuencias adversas.

Como es de suponerse, en los bloqueos mutuos intervienen los recursos no expropiables.

La secuencia de sucesos que se requiere para usar un recurso es:

1. Solicitar el recurso.
2. Usar el recurso.
3. Liberar el recurso.

3.4.2 Principios de bloqueo mutuo.

Para que haya un bloqueo mutuo, deben cumplirse cuatro condiciones (Coffman 1971):

1. Exclusión mutua: cada recurso está asignado únicamente a un proceso o está disponible.
2. Retener y esperar: los procesos que actualmente tiene recursos que les fueron otorgados previamente pueden solicitar nuevos recursos.
3. No expropiación: no es posible quitarle por la fuerza a un proceso los recursos que le fueron asignados previamente, los debe liberar explícitamente.

4. Espera circular: debe haber una cadena circular de dos o más procesos, cada uno de los cuales está esperando un recurso retenido por el siguiente miembro de la cadena.

Modelado del bloqueo mutuo.

Las cuatro condiciones anteriores pueden modelarse usando grafos dirigidos.

Los grafos tiene dos clases de nodos: procesos (círculos) y recursos (cuadrados). Un arco de un recurso a un proceso indica que dicho recurso está asignado al proceso en cuestión. Un arco de

un proceso a un recurso indica que el proceso está bloqueado esperando ese recurso.

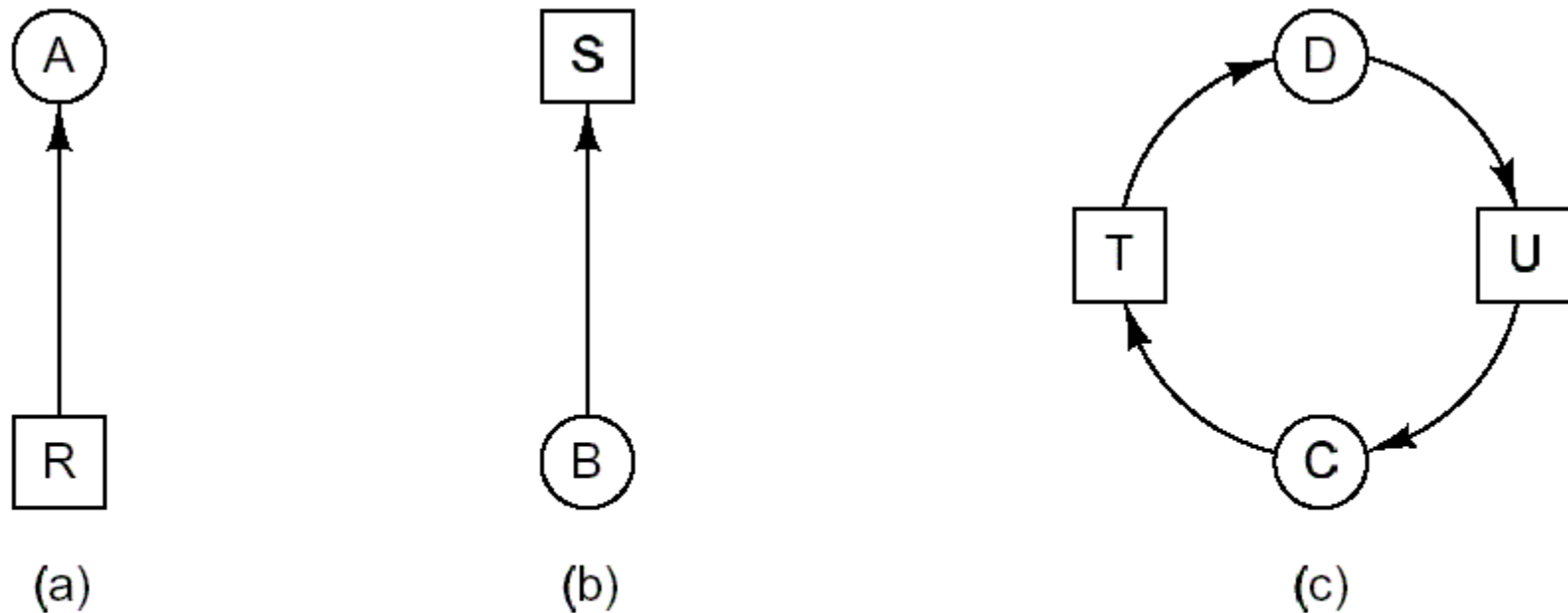


Figure 3-7. Resource allocation graphs. (a) Holding a resource. (b) Requesting a resource. (c) Deadlock.

Un ciclo en el grafo implica que hay un bloqueo mutuo en el que interviene los procesos y recursos del ciclo.

Los grafos de recursos son una herramienta que nos permite ver si una secuencia de petición/liberación dada conduce o no al bloqueo.

Basta con indicar las peticiones y liberaciones paso por paso determinando en cada uno, si el grafo contiene ciclos.

Si el grafo contiene ciclos tenemos un bloqueo mutuo, si no, no hay bloqueo.

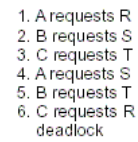


Figure 3-8. An example of how deadlock occurs and how it can be avoided.

3.4.3 Algoritmo del avestruz.

No haga nada.

3.4.4 Detección y recuperación.

Aquí el sistema no hace otra cosa que no sea vigilar las peticiones y liberaciones de recursos. Cada vez que esto pasa, se actualiza el grafo y se determina si contiene algún ciclo.

Si se encuentra uno, se termina uno de los procesos que conforman el ciclo, si esto no rompe el bloqueo mutuo, se termina otro proceso continuando así hasta romper el ciclo. Este esquema es el que comúnmente se usa en las macro computadoras.

3.4.5 Prevención del bloqueo mutuo.

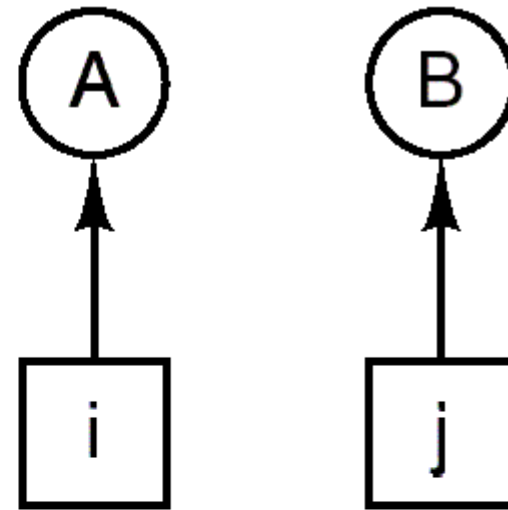
Si se puede asegurar que al menos una de las condiciones de Coffman nunca se satisfaga, el bloqueo mutuo será imposible.

Una forma de evitar la espera circular es crear una numeración global de todos los recursos y aplicar esta regla: los procesos pueden solicitar recursos cuando quieran, pero todas las peticiones deben hacerse en orden numérico y nunca en orden descendente.

Con esta regla, el grafo de asignación de recursos no puede tener ciclos.

1. CD-ROM
2. Printer
3. Plotter
4. Tape drive
5. Robot arm

(a)



(b)

Figure 3-9. (a) Numerically ordered resources. (b) A resource graph.

Aunque el ordenamiento numérico de los recursos elimina el bloqueo mutuo, puede ser imposible encontrar un ordenamiento que satisfaga a todo el mundo.

| Condition | Approach |
|------------------|---------------------------------|
| Mutual exclusion | Spool everything |
| Hold and wait | Request all resources initially |
| No preemption | Take resources away |
| Circular wait | Order resources numerically |

Figure 3-10. Summary of approaches to deadlock prevention.

3.4.6 Algoritmo del banquero para un solo recurso.

Si se cuenta con cierta información por adelantado, se puede evitar el bloqueo mutuo.

El Algoritmo del banquero se debe a Dijkstra.

El algoritmo toma como modelo la forma en que un banquero de una ciudad pequeña podría tratar con un grupo de clientes a los que ha concedido líneas de crédito.

Una lista de los clientes (procesos) junto con el dinero (recursos) se llama **estado** del sistema.

Se dice que un estado es seguro si existe una secuencia de otros estados que conduzca a una situación en la que todos los clientes obtienen préstamos hasta sus límites de crédito.

| Name | Used | Maximum |
|---------|------|---------|
| Andy | 0 | 6 |
| Barbara | 0 | 5 |
| Marvin | 0 | 4 |
| Suzanne | 0 | 7 |

Available: 10

(a)

| Name | Used | Maximum |
|---------|------|---------|
| Andy | 1 | 6 |
| Barbara | 1 | 5 |
| Marvin | 2 | 4 |
| Suzanne | 4 | 7 |

Available: 2

(b)

| Name | Used | Maximum |
|---------|------|---------|
| Andy | 1 | 6 |
| Barbara | 2 | 5 |
| Marvin | 2 | 4 |
| Suzanne | 4 | 7 |

Available: 1

(c)

Figure 3-11. Three resource allocation states: (a) Safe. (b) Safe. (c) Unsafe.

El algoritmo del banquero consiste en considerar cada petición en el momento en el que se presenta y ver si su situación conduce o no a un estado seguro.

Si el estado es seguro, concede lo solicitado, si no, pospone la petición.

3.5 Discos.

Todos los discos reales están organizados en cilindros, cada uno de los cuales tiene tantas pistas como cabezas apiladas verticalmente.

Las pistas se dividen en sectores.

Todos los sectores tiene el mismo número de bytes.

Estos y otros aspectos son administrados por los controladores de discos. Ahora se considerarán algunos aspectos relacionados.

3.5.1 Software de discos.

El tiempo requerido para leer o escribir un bloque de disco está determinado por tres factores:

1. El tiempo de búsqueda (el tiempo que toma mover el brazo al cilindro correcto).
2. El retraso rotacional (el tiempo que tarda el sector correcto en girar hasta quedar debajo de la cabeza).

3. El tiempo de transferencia de datos.

En casi todos los discos el tiempo de búsqueda es mayor que los otros dos, por lo que la idea es reducir este tiempo.

3.6 Algoritmos de planificación del brazo del disco.

Mientras el brazo está realizando una búsqueda para atender una petición, es posible que otros procesos generen otras peticiones y, dependiendo del algoritmo utilizado, será el tiempo de respuesta.

Muchos controladores de disco mantienen una tabla con todas las peticiones pendientes para cada cilindro.

Para los algoritmos siguientes, considérese un disco de 40 cilindros y la siguiente secuencia de peticiones:

11, 1, 36, 16, 34, 9, 12

3.6.1 El primero que llega es el primero que se atiende.

Aquí se aceptan peticiones y éstas son atendidas conforme van llegando, por lo que no hay optimización posible.

¿Cuántos cilindros se recorren en total?

3.6.2 Primero la búsqueda más corta (SSF).

Este algoritmo se base en atender la petición más cercana al cilindro actual a fin de minimizar el tiempo de búsqueda.

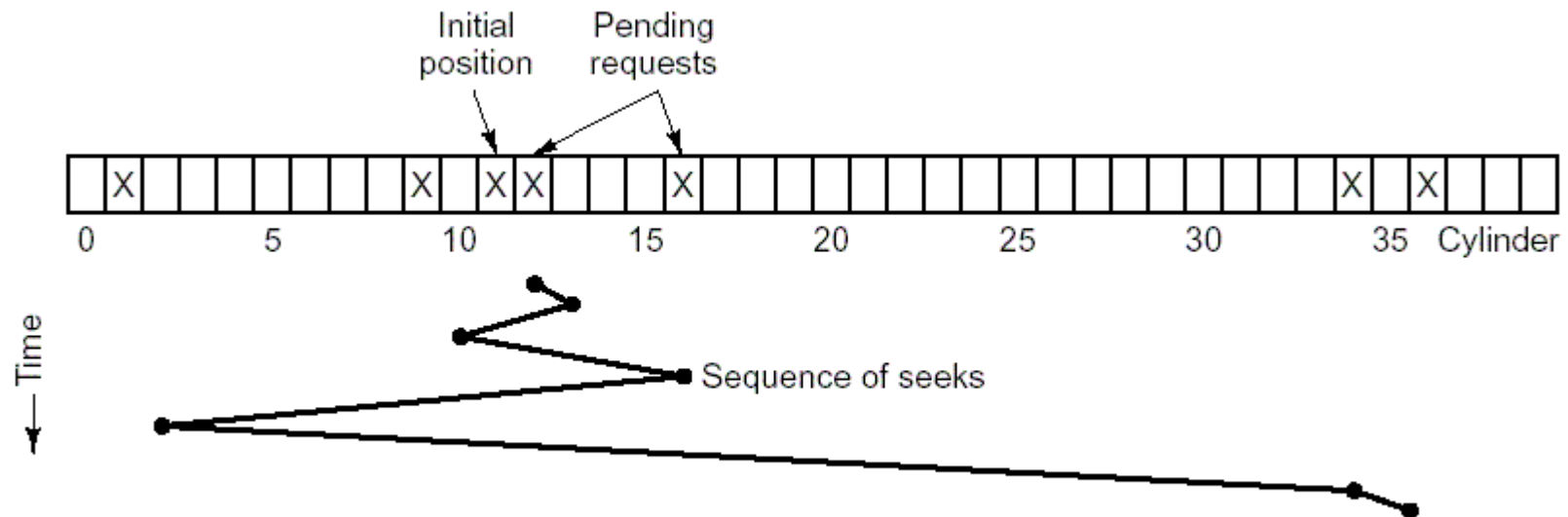


Figure 3-20. Shortest Seek First (SSF) disk scheduling algorithm.

Sin embargo, este algoritmo tiene el problema de que si la carga del disco es elevada, el brazo tenderá a permanecer en la parte media del disco la mayor parte del tiempo afectando significativamente a las peticiones de los extremos.

3.6.3 Algoritmo del elevador.

El problema de planificar un elevador de un edificio alto es similar al de planificar un brazo de disco.

Este algoritmo requiere que el software mantenga un bit de dirección: UP o DOWN.

La idea de este algoritmo es moverse en una sola dirección (mientras haya peticiones) e ir atendiendo las peticiones que se encuentre en el camino, para después moverse en la dirección opuesta si hubiera más peticiones por atender.

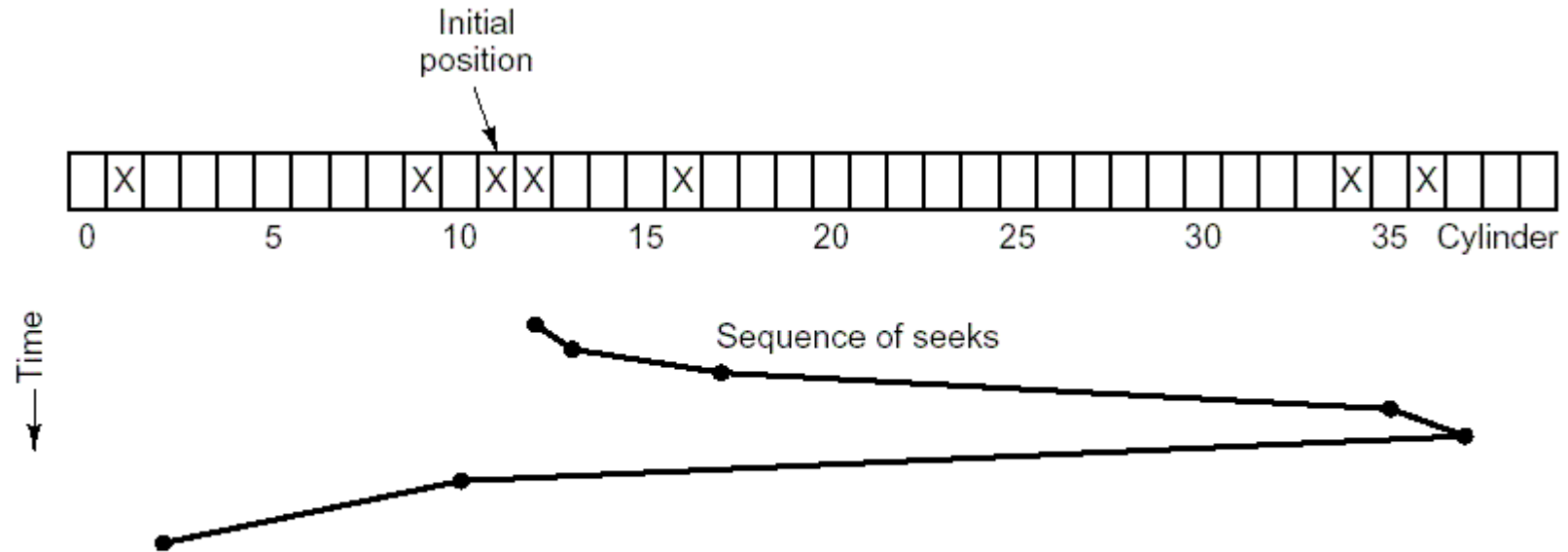


Figure 3-21. The elevator algorithm for scheduling disk requests.

Una propiedad interesante de este algoritmo es que dada cualquier colección de peticiones, el peor caso de recorrido está fijo a dos veces el número de cilindros.