Subsistema de archivos

Objetivo
API
Archivos/directorios
Organización de directorios
Perspectiva desde un proceso
Sistemas de archivos: Bloques secuenciales, enlazados y FAT; bloques de índices, í-nodos de Un*x.
Estrategias de asignación de bloques
Integridad del sistema de archivos

Objetivo

- Implementa el almacenamiento de información a largo plazo
- Provee mecanismos para:
    - Persistencia de datos
    - Compartir información
- La *unidad* de almacenamiento es el *Archivo.*

El kernel debe proveer ciertas operaciones sobre los archivos (ejemplo api unix)

- Creación y Eliminación
- Lectura y Escritura, ¿Copiado?
- Truncar un archivo

- Administrar automáticamente el almacenamiento secundario correspondiente.
- Permitir referenciar a los archivos mediante un nombre simbólico.
- Proteger la integridad de los archivos ante fallas del sistema.
- Permitir la compartición de archivos entre procesos cooperativos, pero evitar el acceso no autorizado. (Con permisos)

Archivos y directorios

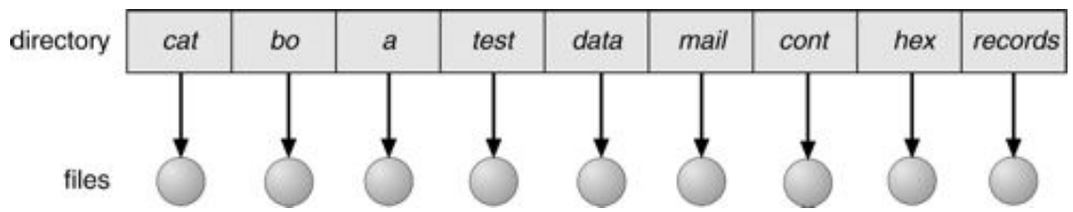
- Archivos: Son los “contenedores” de la información. Pueden tener estructura o no.

- Directorios: Proveen a los archivos de un nombre simbólico para comodidad del usuario.

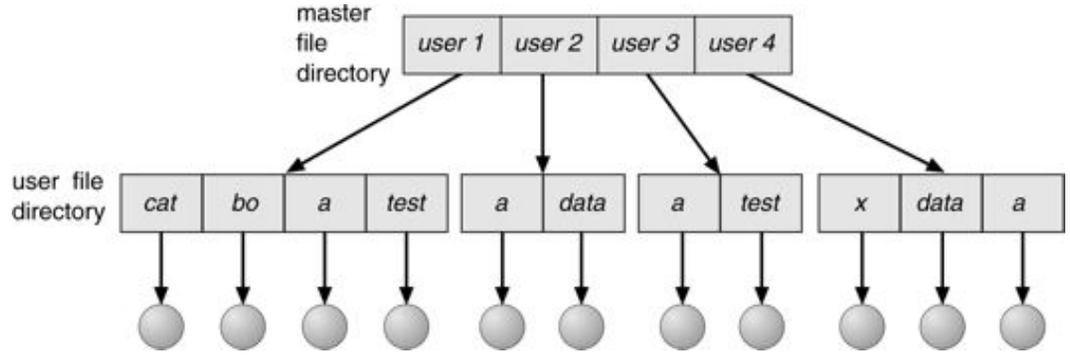
- Ambas estructuras tienen que estar en almacenamiento secundario. El kernel abstrae el dispositivo involucrado.

Organización de directorios

- Un nivel

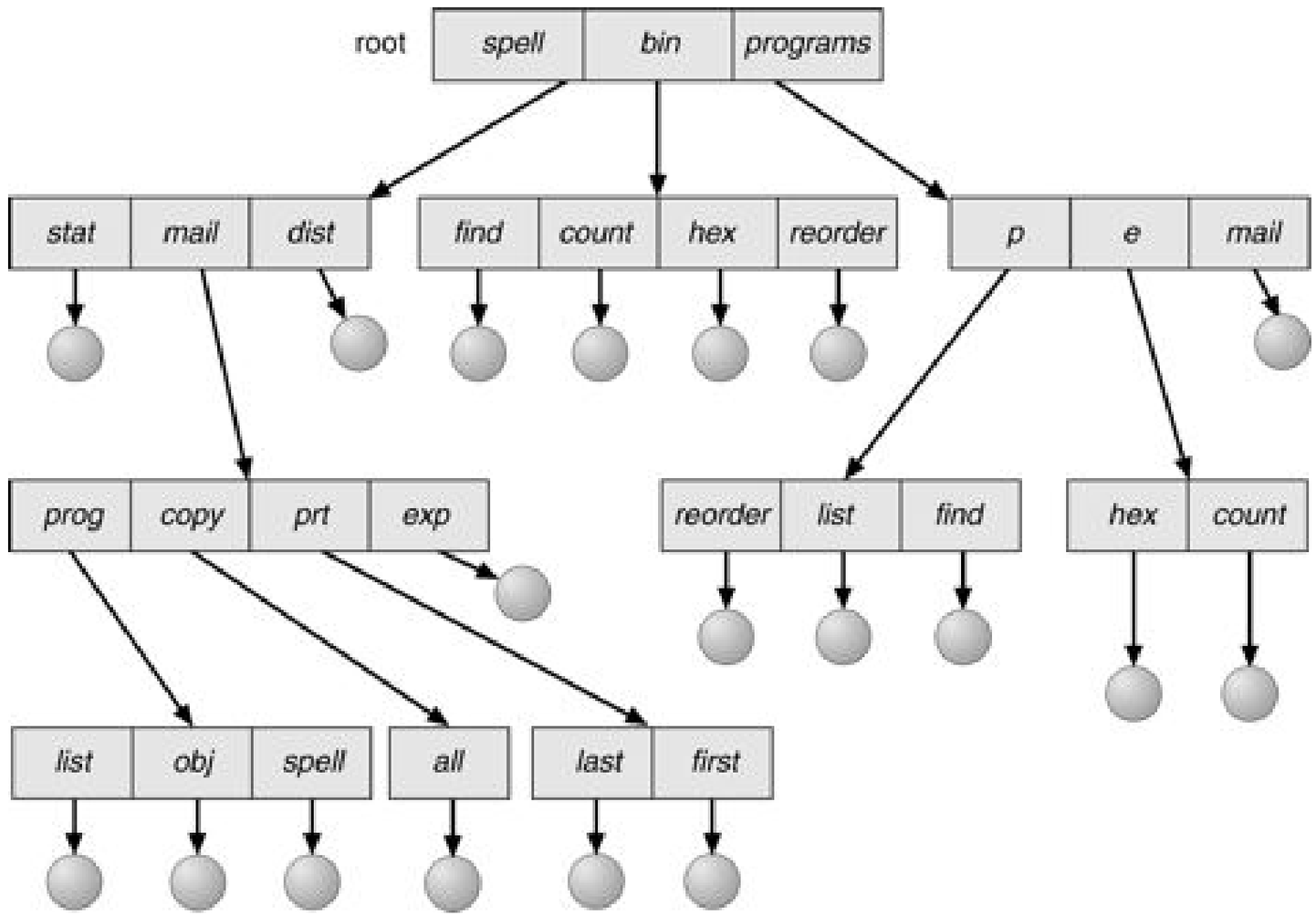


- Dos niveles



Organización de directorios (cont.)

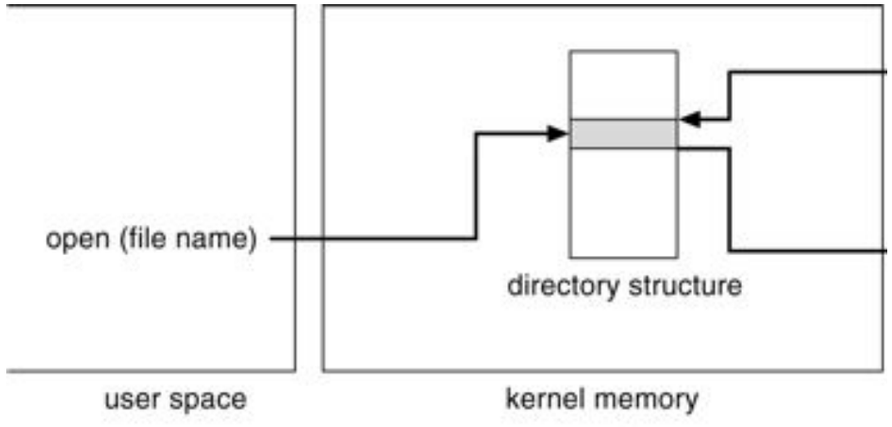
- Árbol de directorios



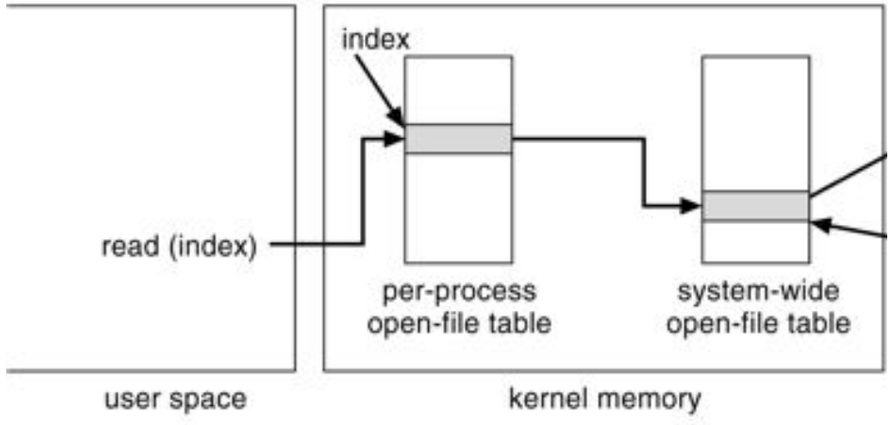
Perspectiva desde un proceso

- Un proceso tiene asignado un *directorio actual.* Las referencias a archivos que no son totalmente especificadas son relativas a éste.

- Un proceso tiene una tabla de *descriptores de archivos .* Las solicitudes de un proceso se hacen relativas a estos descriptores.



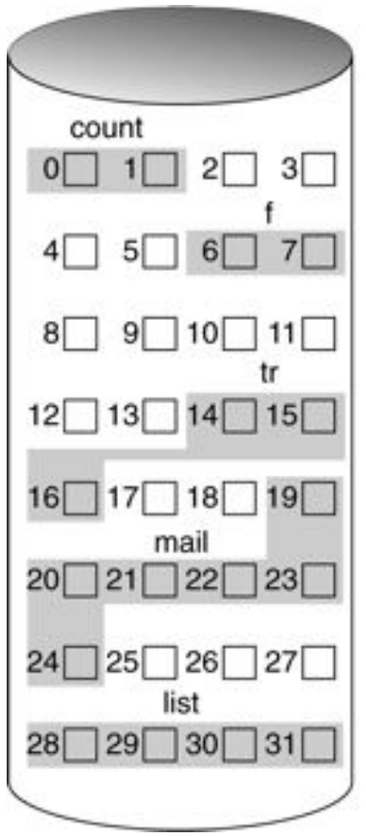
(a)



Sistemas de archivos

Organización física de un archivo en disco

- El disco se divide en *bloques* de tamaño fíjo. El contenido de un archivo se “reparte” en los bloques.
  - Muchos esquemas posibles. Ejemplo más sencillo: secuencialmente.

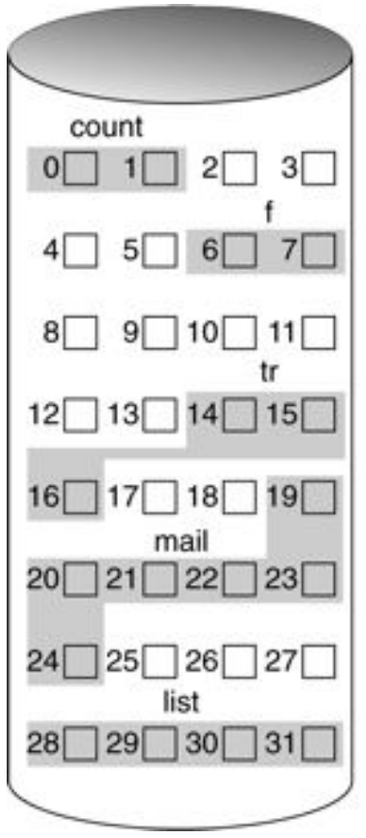


directory

file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Bloques secuenciales

- Pros:
  - Acceso aleatorio.
  - Un bloque dañado sólo daña un archivo.
- Cons: Fragmentación => Compactación.
  - Difícil añadir o eliminar bloques bloques al interior del archivo.
  - Hay que saber de antemano la cantidad de bloques que hay que asignar.



directory

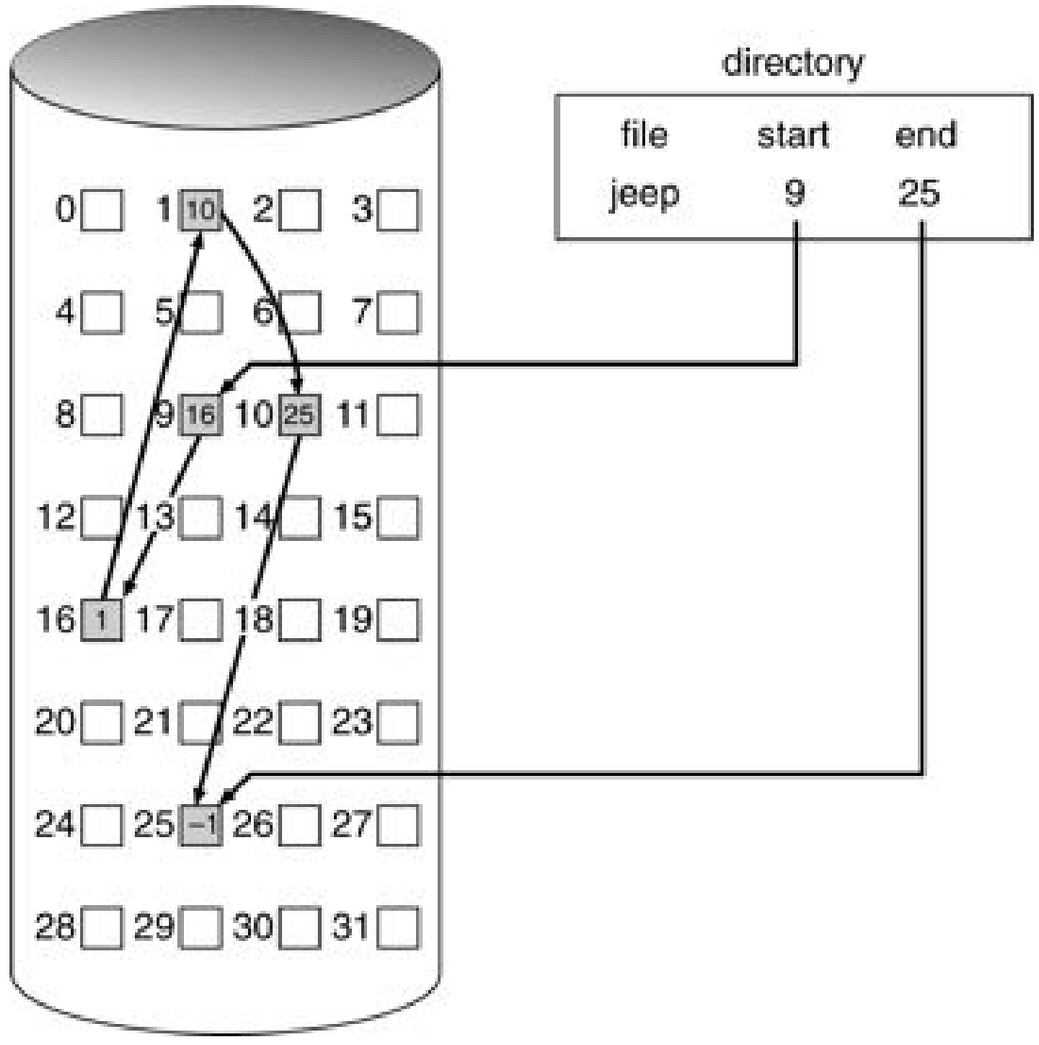
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

Bloques enlazados

- Los bloques de un archivo se estructuran en una lista ligada contenida en él mismo.

- En el directorio se guarda un apuntador al primero y último bloque.

- Cada bloque tiene un apuntador al siguiente.



Bloques enlazados

- Pros: No hay fragmentación interna.
    - No hace falta saber el tamaño del archivo de antemano.
    - No hace falta realizar compactación. (aunque ayuda: defrag)
- Cons:
    - Acceso secuencial únicamente. Avanzar al bloque siguiente implica una lectura de disco.
    - Sobrecarga de espacio por los apuntadores.
- Para minimizar las desventajas, se usan *clusters* en lugar de bloques: conjuntos de tamaño fijo de bloques consecutivos.

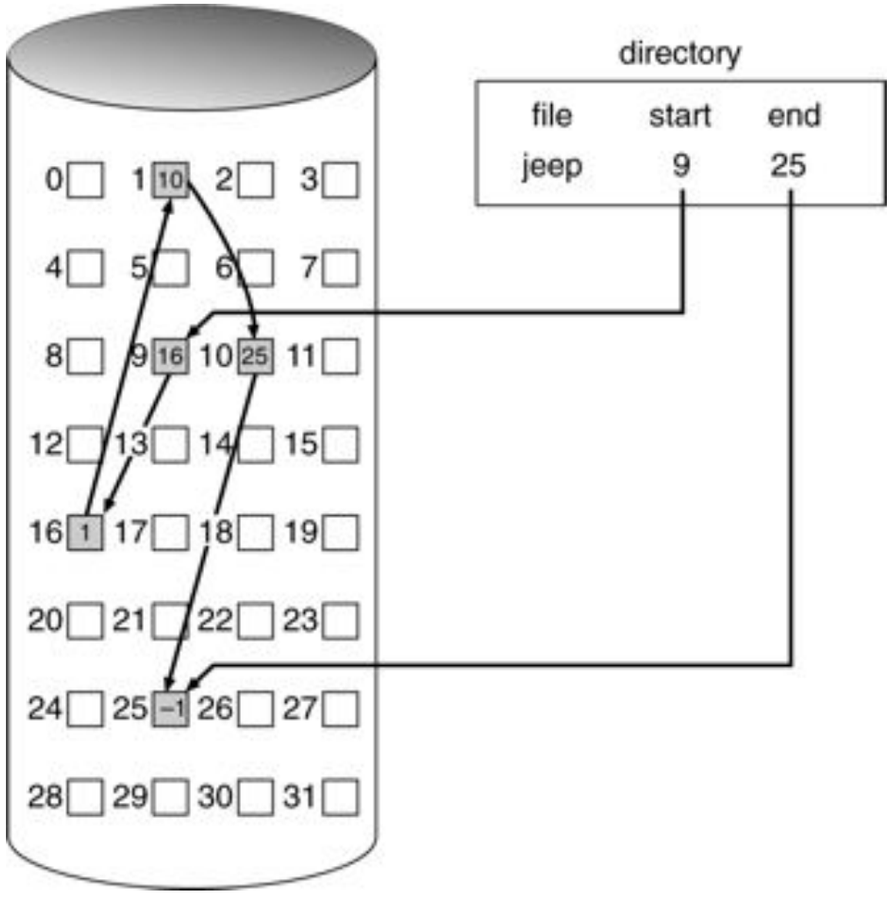
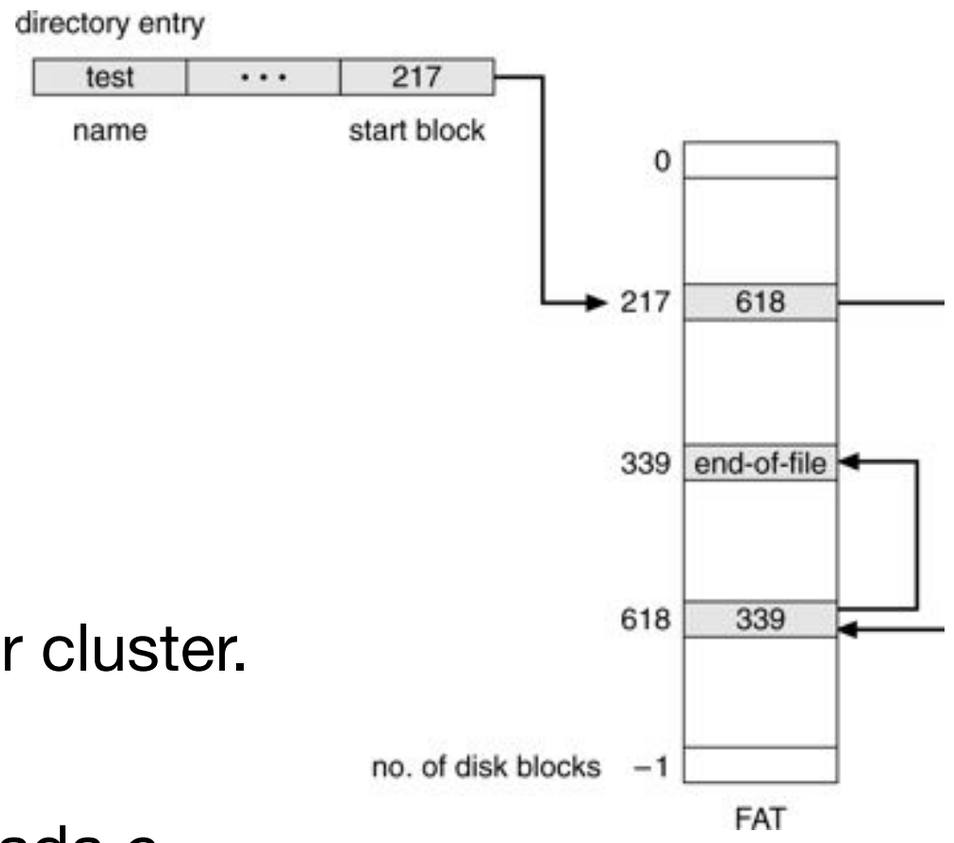
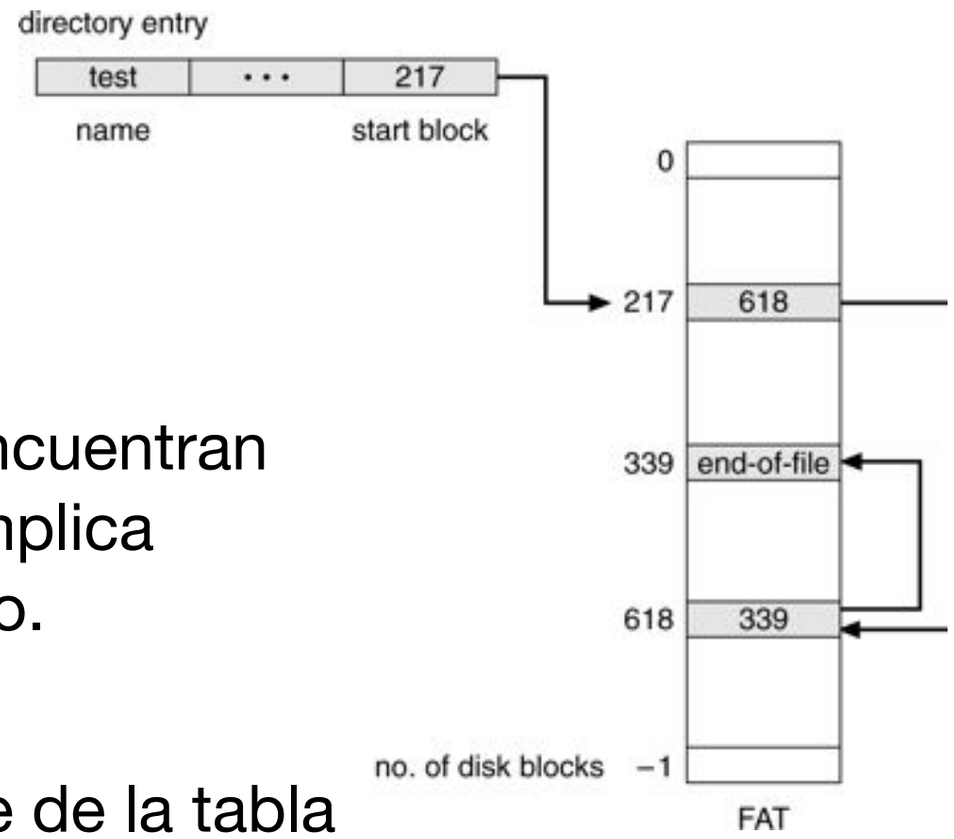


Tabla de asignación de archivo (FAT)



- Variación del esquema enlazado.
- El disco se divide en clusters.
- Se hace UNA tabla que contiene una entrada por cluster.
- Si el cluster *c* es el *i*-ésimo del archivo *f*, la entrada *c*-ésima de la tabla contiene el número del cluster *(i+1)*-ésimo de *f.*
- Una entrada de directorio contiene un apuntador al primer y último cluster del archivo.

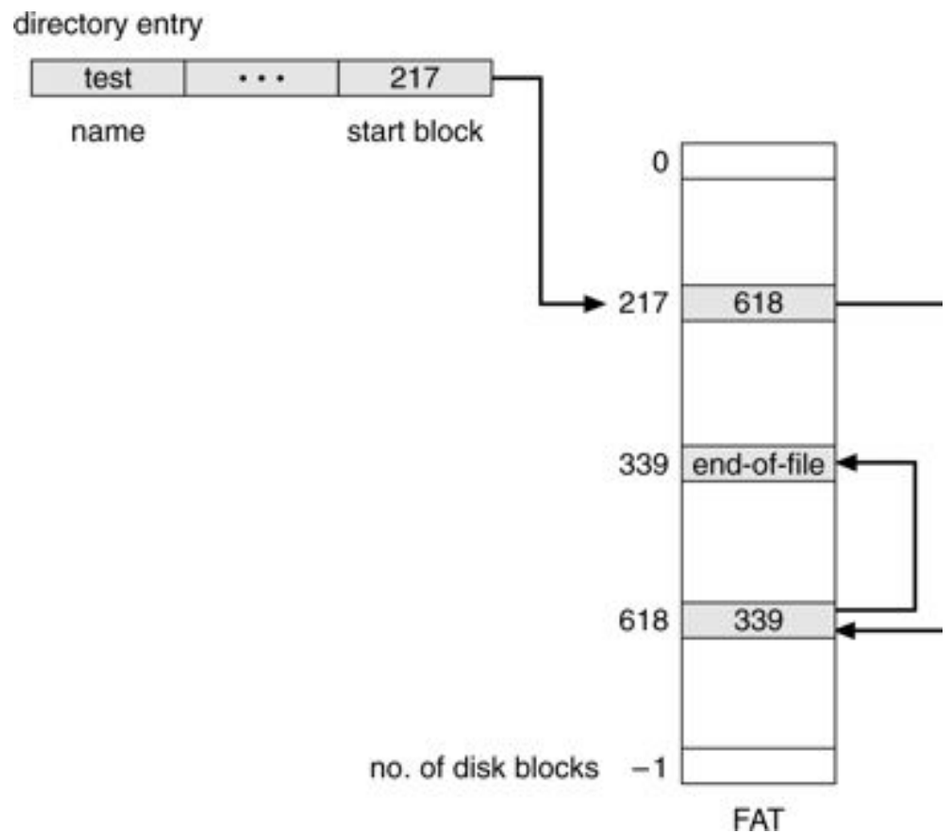
Tabla de asignación de archivo (FAT)



- Variación del esquema enlazado.
- La ventaja es que TODOS los apuntadores se encuentran en bloques consecutivos. Seguir apuntadores implica mucho menor movimiento de la cabeza del disco.
- La desventaja es que la corrupción de un bloque de la tabla puede invalidar MUCHOS archivos.
  - Hay que mantener múltiples copias de la FAT para aumentar la tolerancia a fallos.
- Se usa muchísimo: MsDOS, Windows 9x, Tarjetas de memoria, Discos USB, Cámaras, ...

Tabla de asignación de archivo (FAT)

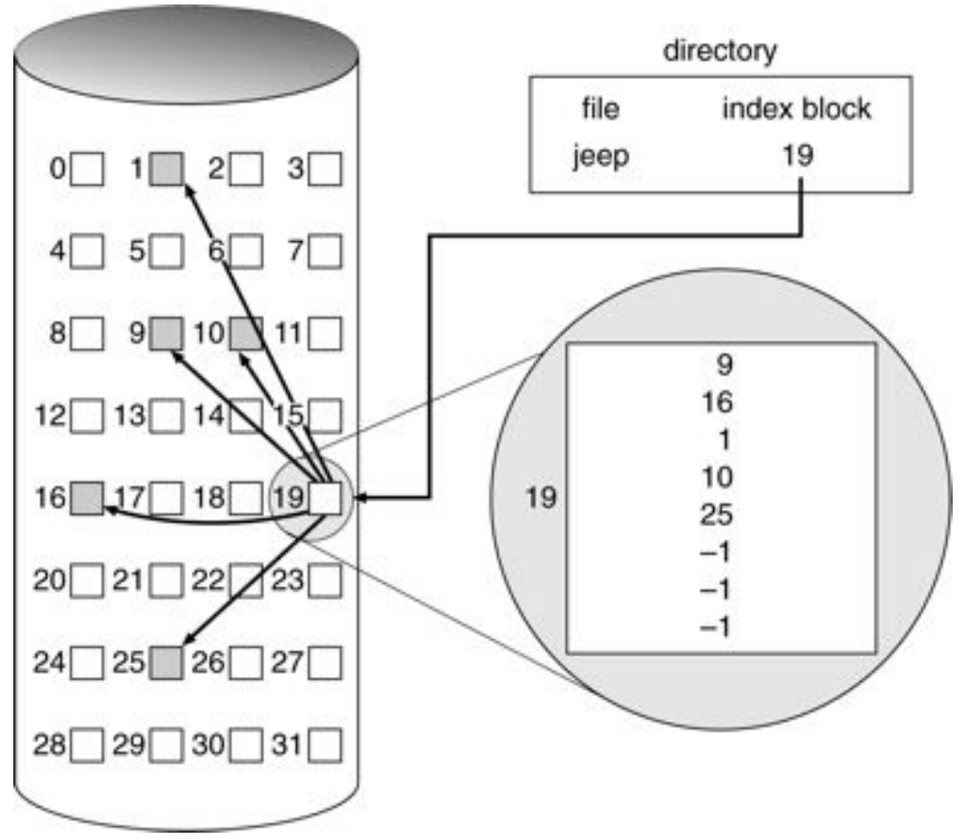
- Implementaciónes estándar:
  - FAT12, FAT16, FAT32.



Bloque de índices

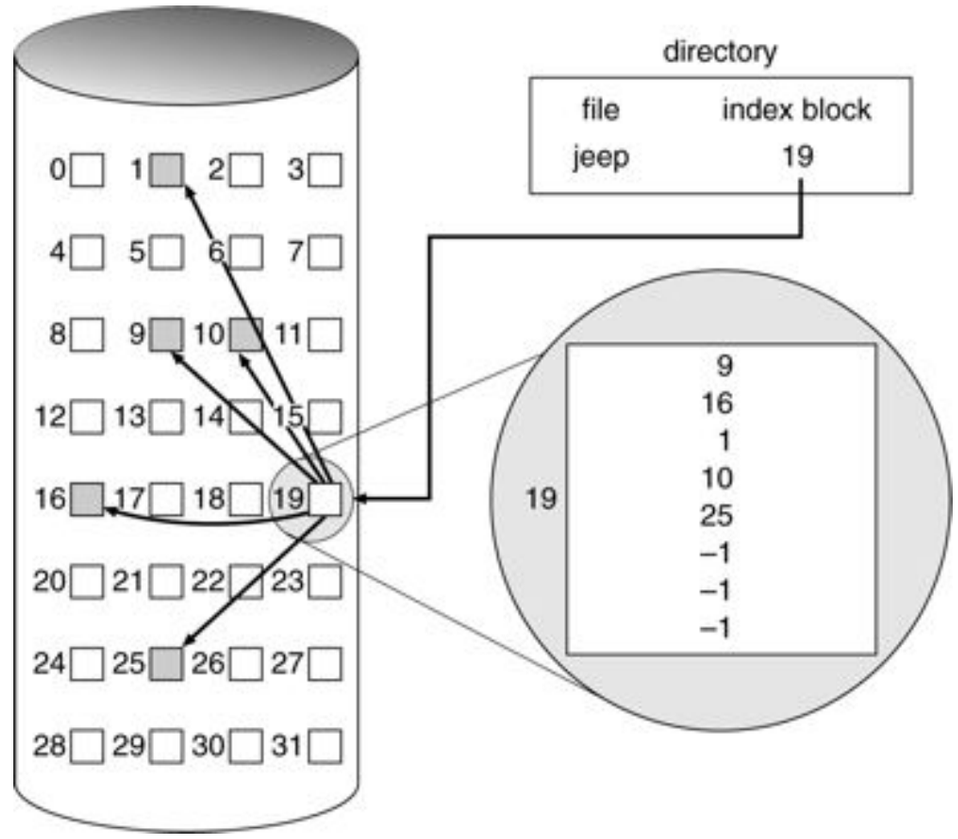
- Un bloque especial juega la función de *arreglo de apuntadores* a los bloques del archivo.

- Es más o menos análogo al mecanismo de *paginación* en la memoria virtual.



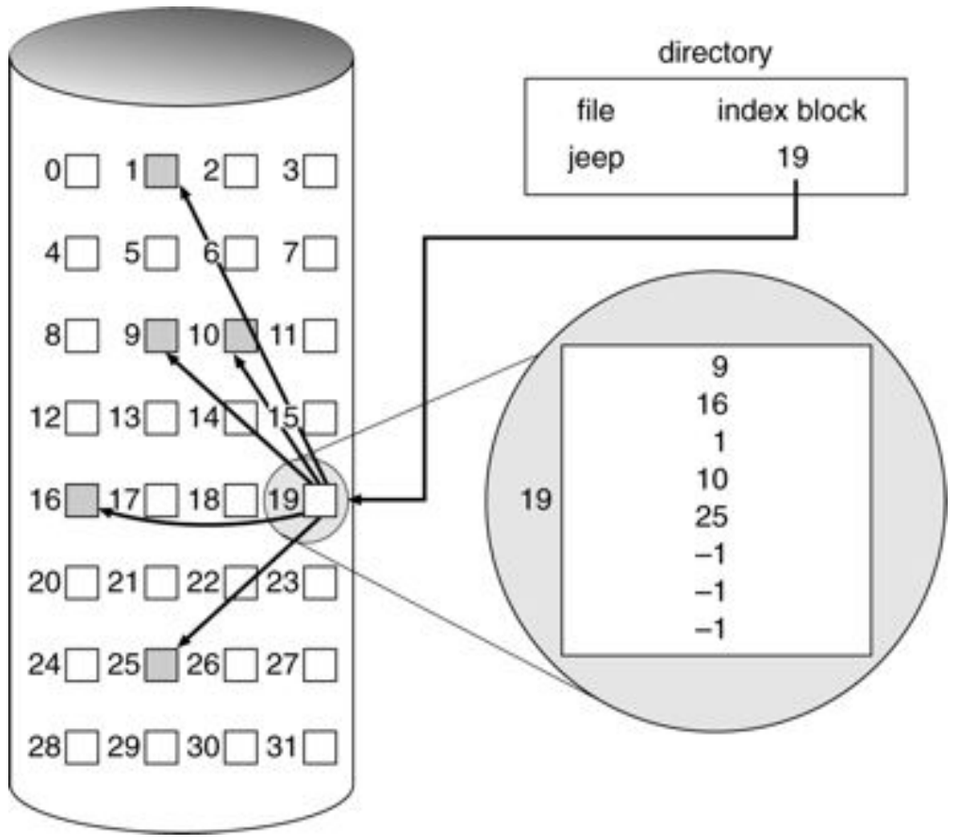
Bloque de índices

- Pros:
  - Acceso eficiente (aleatorio) al archivo.
  - Un bloque dañado solo afecta a un archivo.
- Cons:
  - Desperdicia espacio en archivos pequeños.
- ¿Qué tamaño debe tener el bloque de índices?



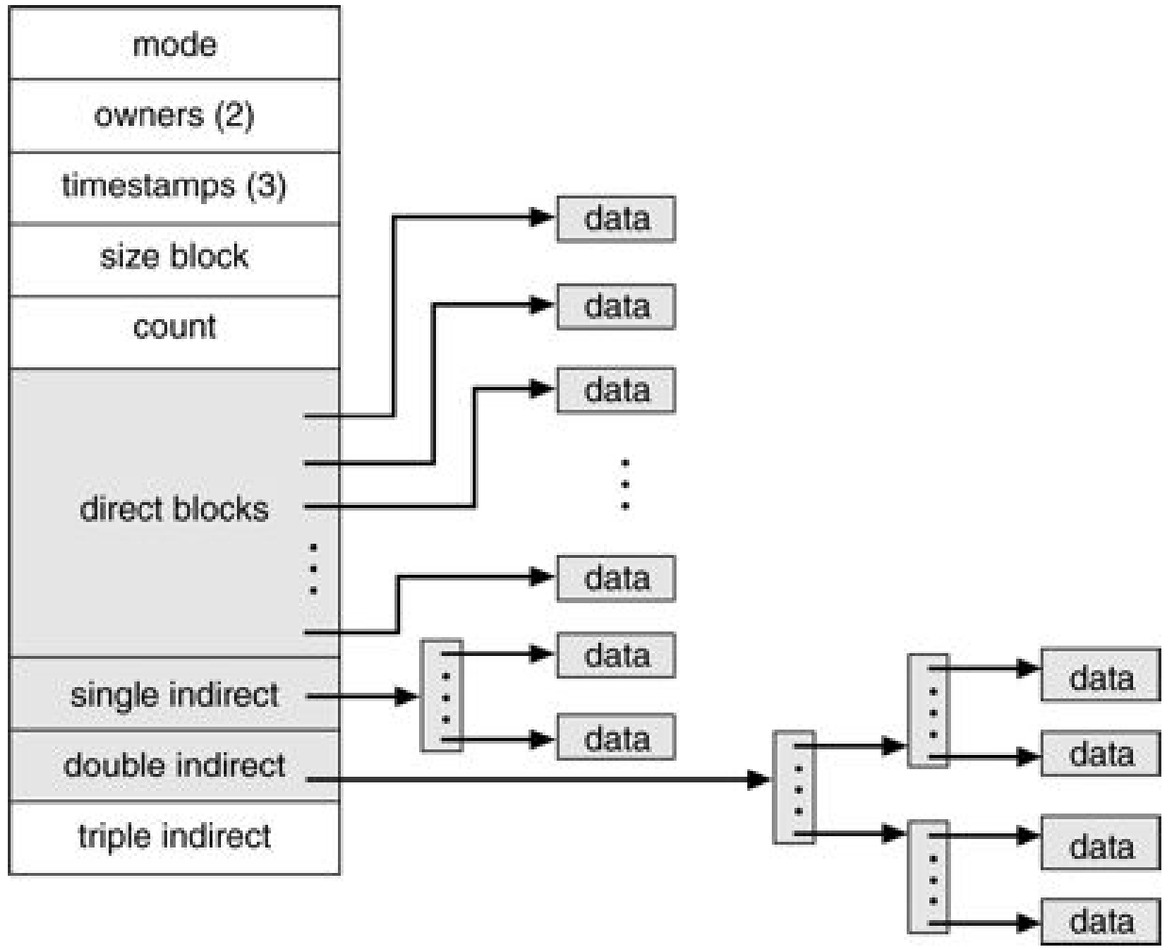
- ¿Qué tamaño debe tener el bloque de índices?

- Múltiples estrategias para no depender del tamaño de bloque:
  - Esquema enlazado: Una lista ligada de bdi.
  - Esquema multinivel: Una jerarquia de dos o más niveles, similar al esquema de directorio/tabla en paginación.
  - Esquema combinado...



Esquema Combinado

- El *i-nodo* contiene unos cuantos bloques *directos*.
    - Con esto se accesan los primeros bloques, y es suficiente para archivos pequeños.
- También contiene una entrada a bloque indirecto, una a dobles indirecto, y una a triple indirecto.
    - Estás entradas se van usando conforme el tamaño del archivo va aumentando.
- Los apuntadores se pueden cachear en memoria, pero se *dispersan* en el disco.
- En general es buena estrategia, y la usada por la mayoría de los UNIX.

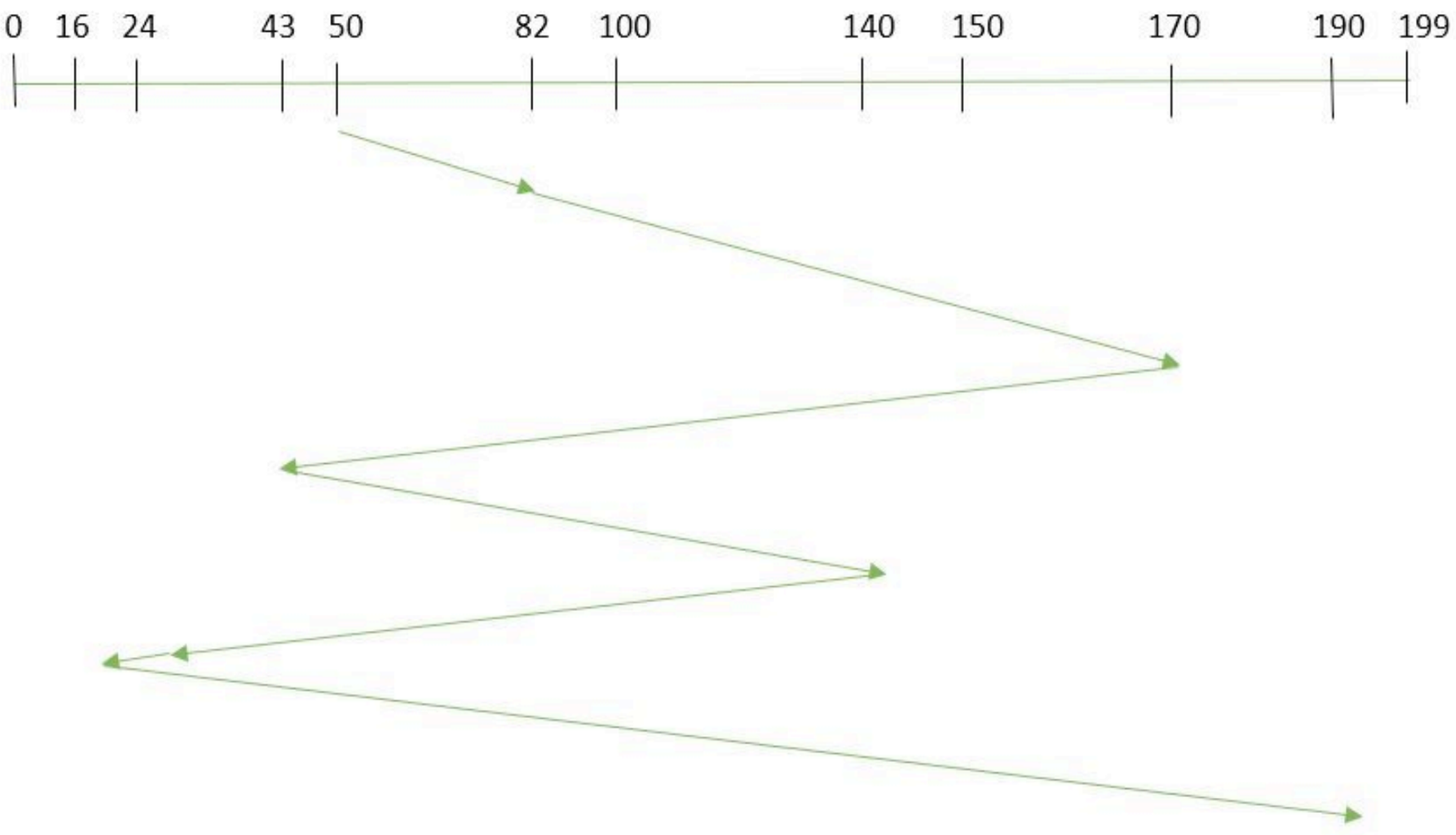


Algoritmos de calendarización de disco

- FCFS
- SCAN (tren)
    - CSCAN
- LOOK (elevador)
    - CLOOK

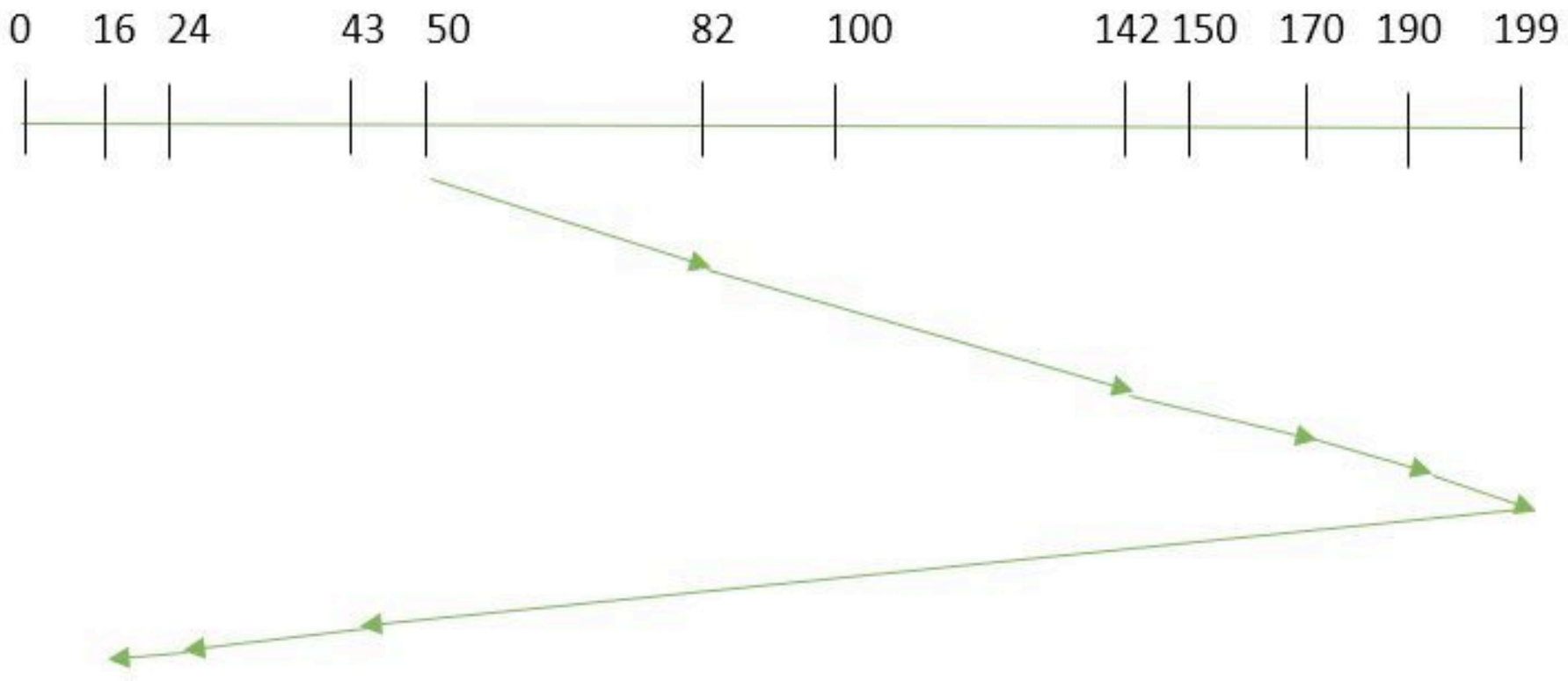
FCFS: First come, first served

- Secuencia de solicitudes: 82,170,43,140,24,16,190
  Posición actual: 50



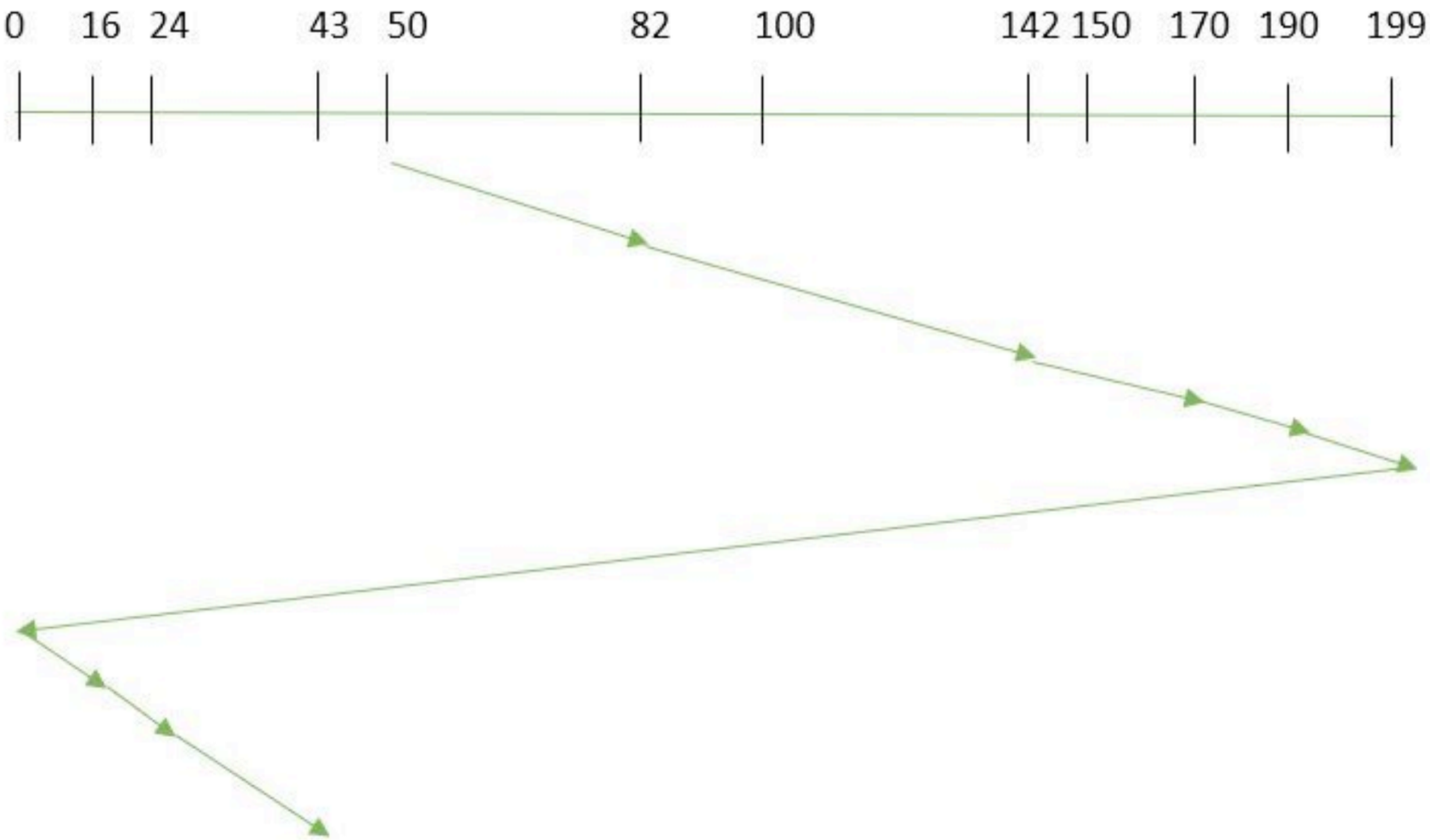
SCAN: Ida y vuelva a ambos extremos y dejando solicitudes al pasar por su destino

- Secuencia de solicitudes: 82,170,43,140,24,16,190
  Posición actual: 50



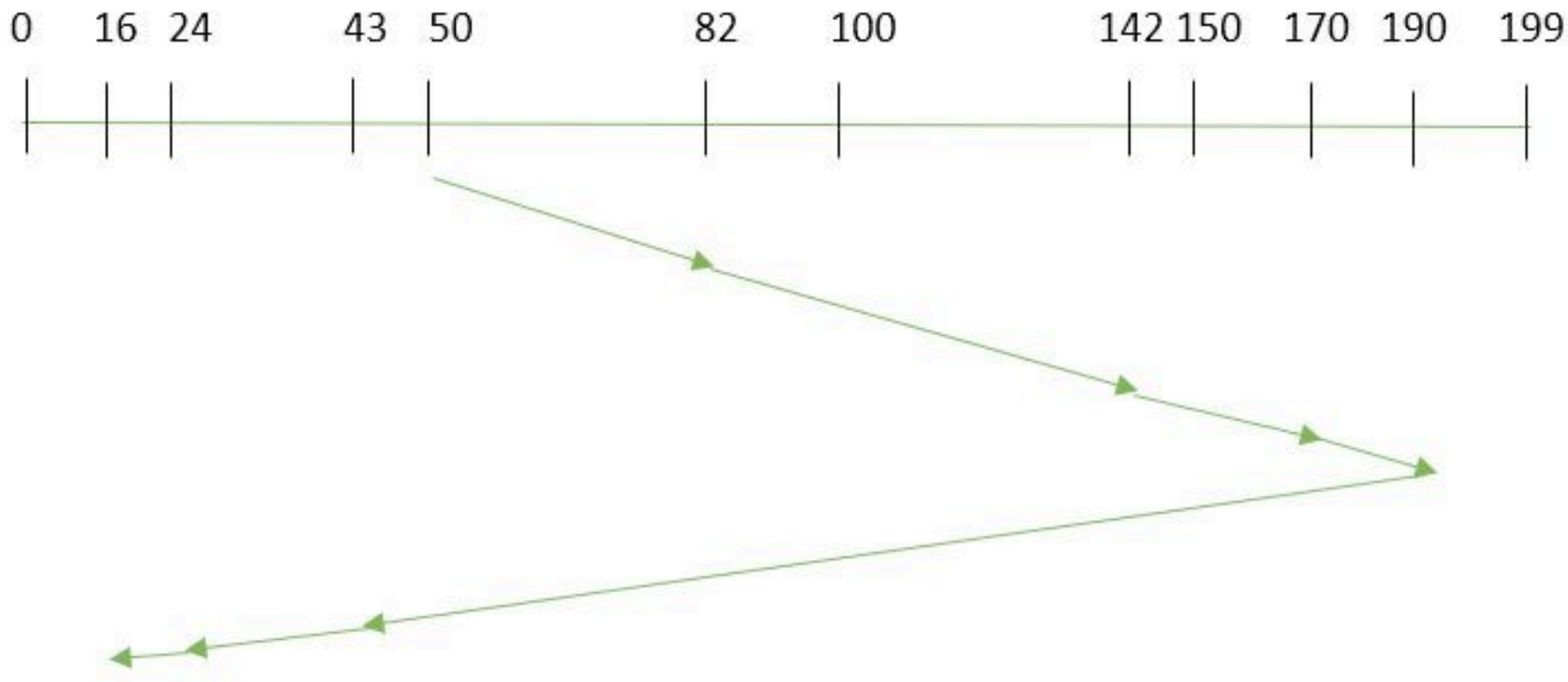
CSCAN: Movimiento como en SCAN, pero dejando solicitudes sólo en una dirección

- Secuencia de solicitudes: 82,170,43,140,24,16,190
  Posición actual: 50



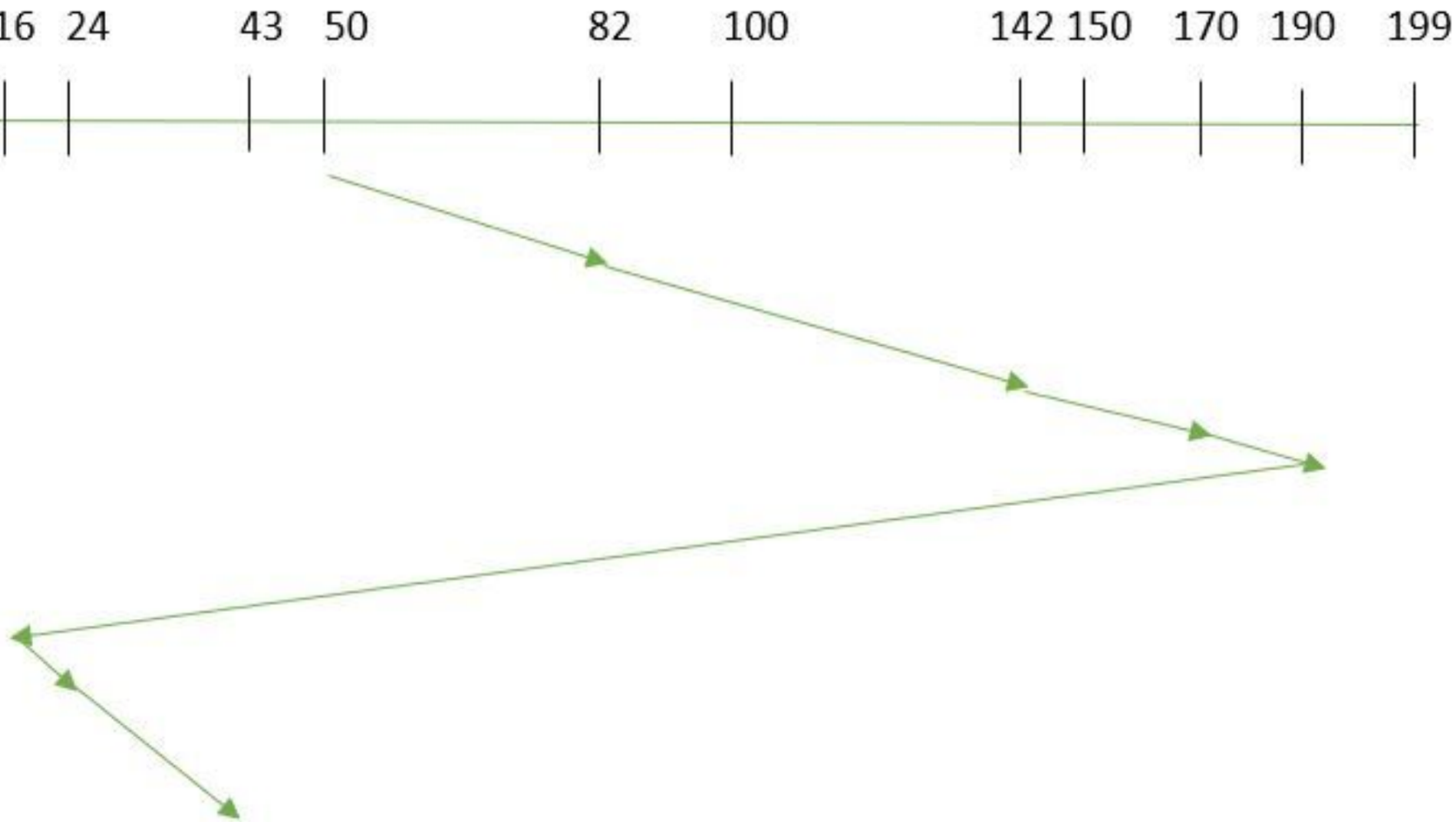
LOOK: Como un elevador. Avanza en una dirección mientras haya una solicitud pendiente. Si no la hay, cambia de dirección.

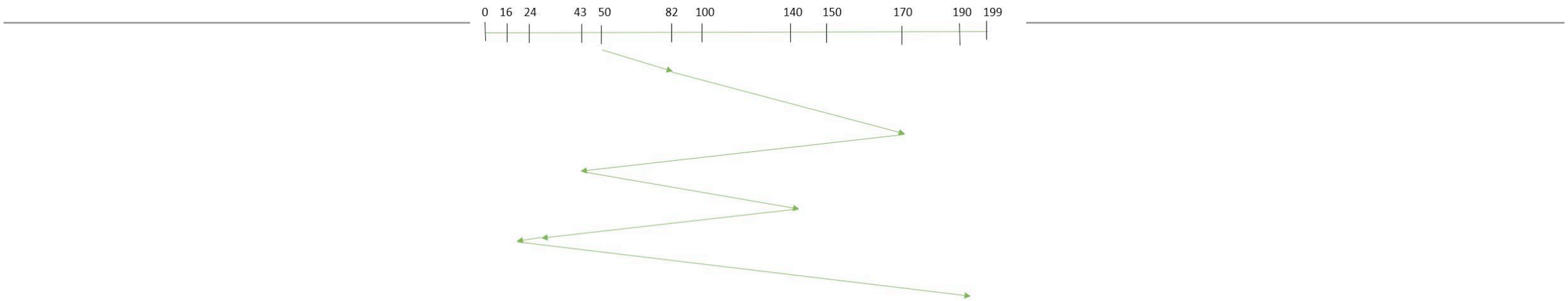
- Secuencia de solicitudes: 82,170,43,140,24,16,190
  Posición inicial: 50

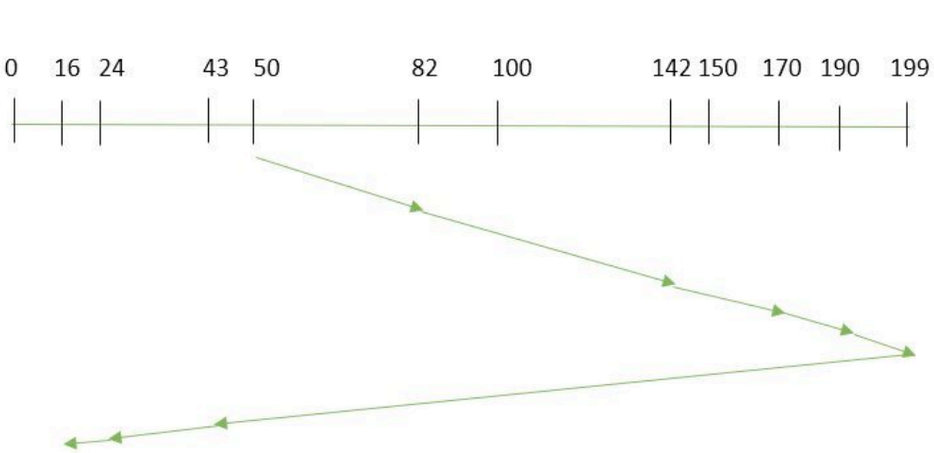


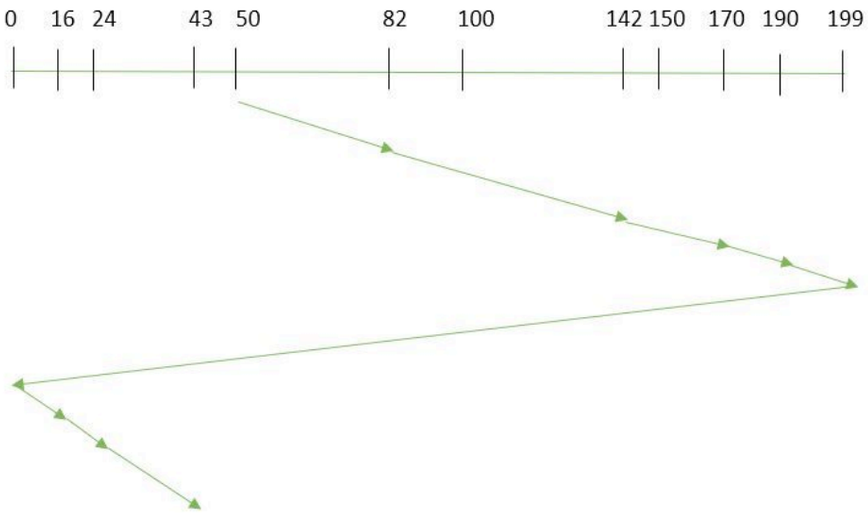
CLOOK: Movimiento como en LOOK, pero sólo se sirven solicitudes en una dirección.

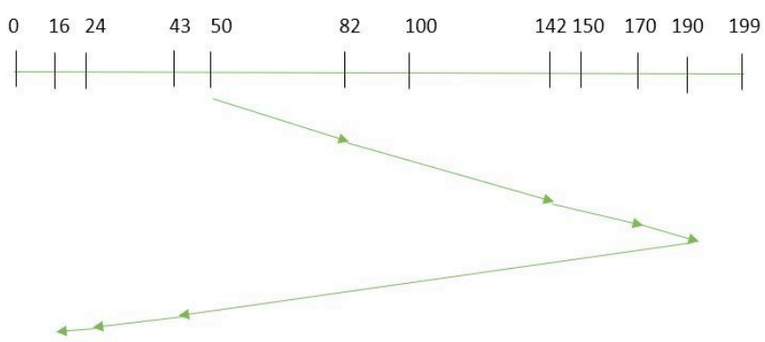
- Secuencia de solicitudes: 82,170,43,140,24,16,190
  Posición inicial: 50

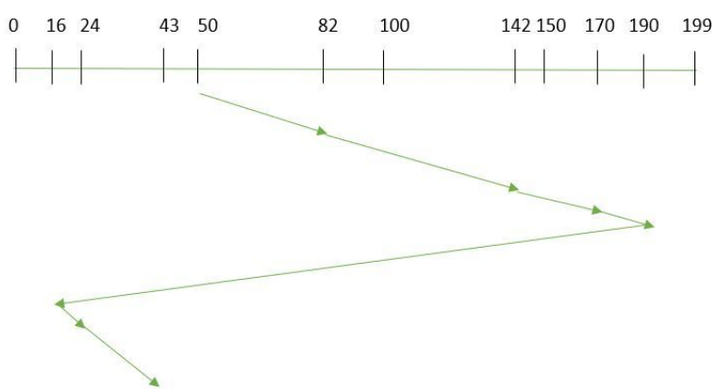



0 16 24 43 50 82 100 140 150 170 190 199


0 16 24 43 50 82 100 142 150 170 190 199


0 16 24 43 50 82 100 142 150 170 190 199


0 16 24 43 50 82 100 142 150 170 190 199


0 16 24 43 50 82 100 142 150 170 190 199

Integridad del sistema de archivos

Integridad

- Probablemente la componente más valiosa de un sistema es la información que contiene. ¡Es muy importante protegerla!

- Primer paso: múltiples copias de estructuras críticas del sistema de archivos. P.Ej. FAT.

- Idealmente el sistema de archivos soporta un buen esquema de respaldos.

- (cada que el sistema de archivos se restaura, se puede aprovechar la operación para compactar y desfragmentar los archivos.)

    - Respaldo periódico (o masivo):

        - Se lleva a cabo de manera periódica. Todo el sistema de archivos es respaldado cada vez.

    - Visión potencialmente inconsistente de los datos. Lo ideal es hacer un *snapshot*, respaldar el snapshot, y luego liberarlo.

    - Respaldo incremental…

Respaldo incremental

- Sólo se copia la información que no aparece en el último respaldo.

- Cada *entrada del directorio* se extiende con un bit “sucio” que es puesto a 1 cuando el archivo cambia. El bit se reinicia a 0 cuando el archivo se respalda.

- Cada *directorio* tiene un bit que indica si alguno de sus archivos o subdirectorios debe respaldarse.

- Contras: el procedimiento de restauración es mucho más complicado.

Sistemas de archivos transaccionales

- Agilizan la verificación y restauración de sistemas de archivos

- Mismas ideas que transacciones en bases de datos

    - Se utiliza un a bitácora en donde se indican las operaciones a realizarse, antes de realmente llevarlas a cabo, junto con las operaciones que las pueden deshacer.

        - Con la bitácora, se puede finalizar (commit) o cancelar (rollback) una transacción.

- Actualmente todos los sistemas de archivos principales son transaccionales: NTFS, EXT4, UFS/FFS, y por supuesto los más modernos: BTRFS, APFS y ZFS.

- La notable excepción de un sistema de archivos usado ámpliamente es FAT.