# Programa

* Es estático
* No tiene program counter
* Existe desde que se edita hasta que se borra

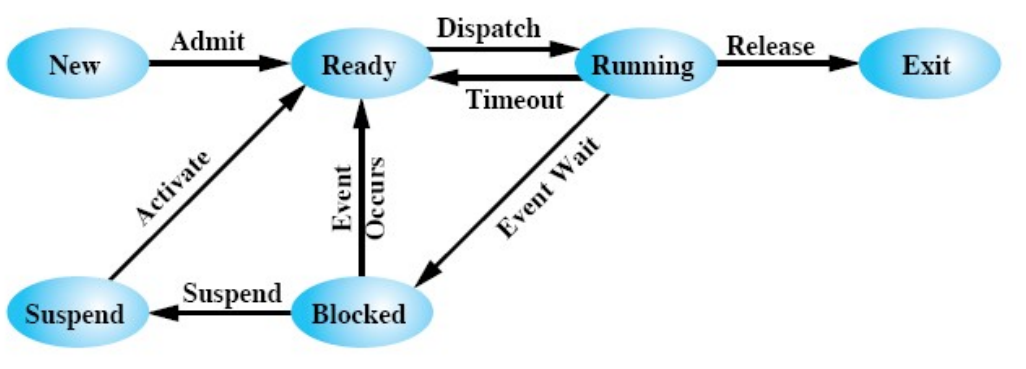
# Proceso

* Programa en ejecución
* Tarea, Job y Proceso son lo mismo
* Según su historial de ejecución se pueden clasificar en
  + CPU Bound
  + I/O Bound
* Es dinámico
* Tiene program counter
* Su ciclo de vida comprende desde que se lo ejecuta hasta que termina

## PCB

* Una por proceso
* Contiene información del proceso
* Es lo primero que se crea cuando se hace un fork y lo último que se borra cuando termina

## Estados



# Planificadores

* Es la clave de la multiprogramación
* Está diseñado de manera apropiada para cumplir las metas de
  + Menor tiempo de respuesta
  + Mayor rendimiento
  + Uso eficiente del procesador

Long Term Scheduler

* Admite nuevos procesos a memoria
* Controla el grado de multiprogramación

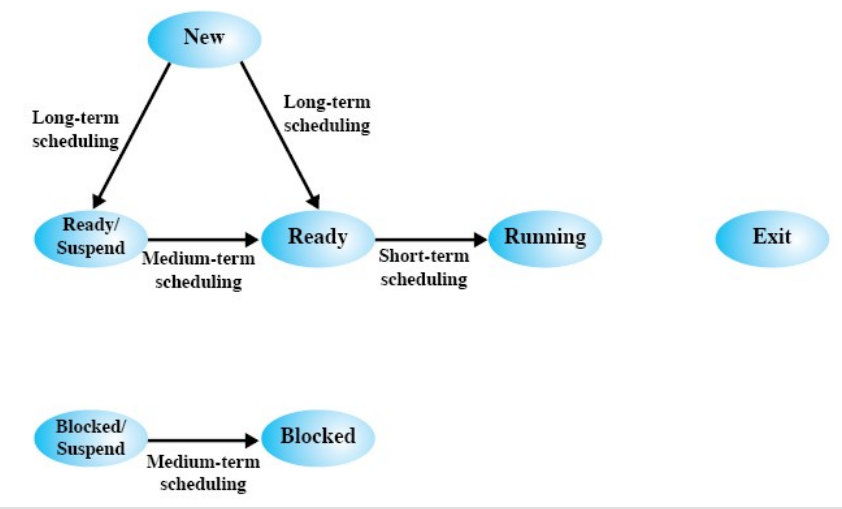
Medium Term Scheduler

* Realiza el swapping entre disco y la memoria cuando el SO lo determina

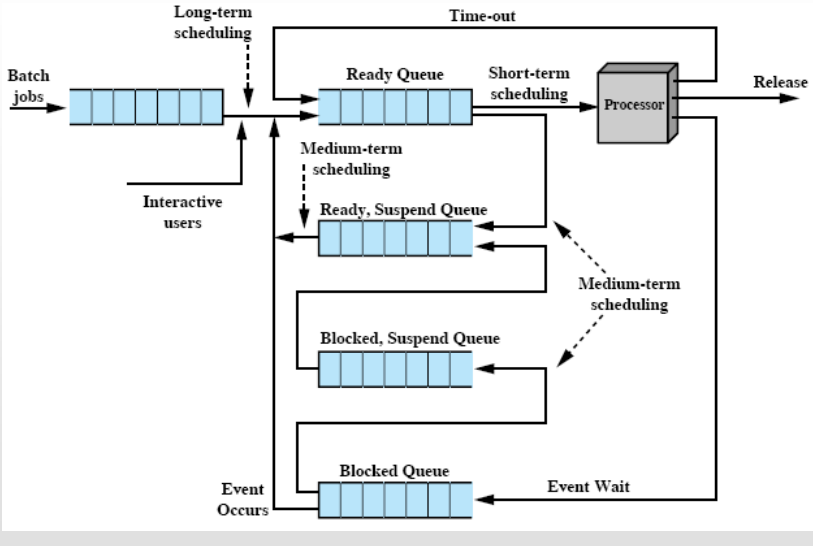
Short Term Scheduler

* Determina qué proceso pasará a ejecutarse

## Planificadores y estados



## Planificadores y Colas



# Tiempos de los procesos

* Retorno:
  + Tiempo que transcurre desde que el proceso llega al sistema hasta que completa su ejecución
* Espera:
  + Tiempo que el proceso se encuentra en el sistema esperando (sin ejecutarse)
  + Tiempo de Retorno – Tiempo usando la CPU
* Promedios:
  + Tiempos promedios de los anteriores

# Políticas apropiativas y no apropiativas

* No apropiativo
  + Una vez que el proceso está en estado de ejecución, continúa hasta que termina o se bloquea por algún evento (EJ: I/O)
* Apropiativa
  + El proceso en ejecución puede ser interrumpido y llevado a la cola de listos
    - Mayor overhead pero mejor servicio
    - Un proceso no monopoliza el procesador

# Algoritmos de planificación

## First In First Out (FIFO)

* No apropiativo
* Cuando hay que elegir un proceso para ejecutar, se selecciona el más viejo (el primero en llegar)
* No favorece a ningún tipo de procesos pero en principio se puede decir que los CPU Bound terminan al comenzar su primer ráfaga, mientras que los I/O Bound

## Shortest Job First (SJF)

* Política no apropiativa que selecciona el proceso con la ráfaga más corta
* Cálculo basado en la ejecución previa
* Procesos cortos se colocan delante de procesos largos
* Los procesos largos pueden sufrir inanición

## Round Robin (RR)

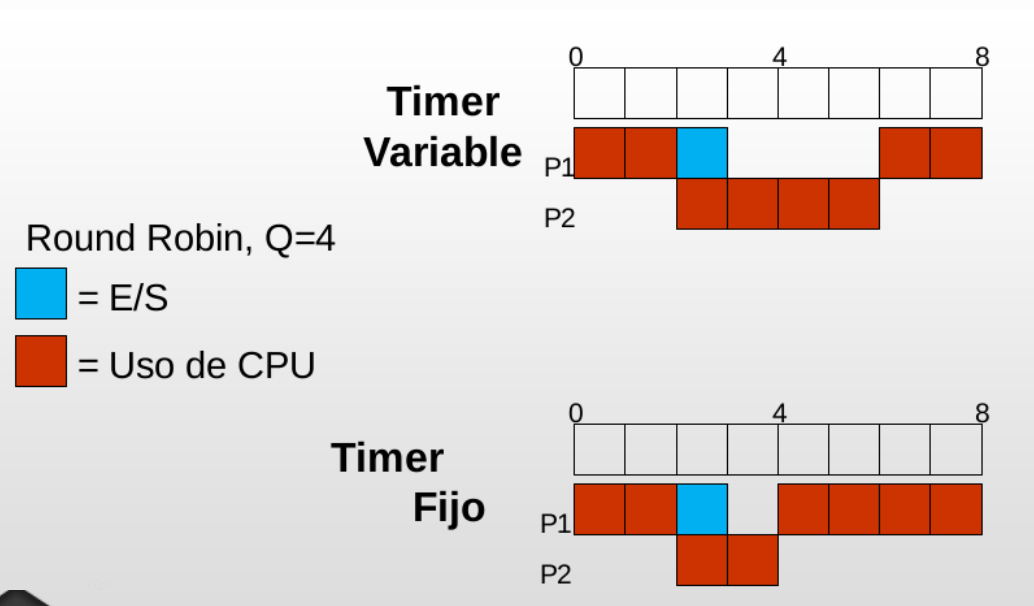
* Política basada en un reloj
* Apropiativa
* Quantum (Q)
  + Medida que determina cuánto tiempo podrá usar el procesador cada proceso
  + Un quantum pequeño da mayor overhead de context switch
  + Un quantum grande puede provocar que un proceso no utilice todo su quantum y termine antes de usarlo, por lo que el algoritmo funcionaría como un FIFO
* Cuando un proceso es expulsado de la CPU es colocado al final de la cola de listos y se selecciona otro
* Existe un contador que indica las unidades de CPU en las que el proceso se ejecutó. Si ese contador llega a 0, el proceso es expulsado
  + Este contador puede ser Global (Aplica a todos los procesos) o Local (Almacenado en la PCB)
  + Existen dos variantes con respecto al valor inicial del contador
    - Timer Variable
    - Timer Fijo

### Timer Variable

* El contador se inicializa en Q cada vez que un proceso es asignado a la CPU
* Es el más utilizado

### Timer Fijo

* El contador se inicializa en Q cuando su valor es cero
* Se puede ver como un valor de Q compartido entre los procesos



## Algoritmo con Uso de Prioridades

* Cada proceso tiene un valor que representa su prioridad
  + A menor valor, mayor prioridad
* Se selecciona el proceso de mayor prioridad de los que se encuentran en la cola de listos
* Existe una cola de listos por cada nivel de prioridad
* Procesos de baja prioridad pueden sufrir inanición
  + Solución:
    - Aging:
      * Se incrementa la prioridad de un proceso cuando lleva mucho tiempo esperando en la cola de listos
    - Penalty:
      * Se baja la prioridad de procesos que lleven mucho tiempo usando la CPU
* Puede ser un algoritmo apropiativo como no serlo

## Shortest Remaining Time First (SRTF)

* Versión apropiativa de SJF
* Selecciona al prceso el cual le resta menos tiempo de ejecución en su siguiente ráfaga
* Favorece procesos I/O Bound ya que son los que pasan menos tiempo usando la CPU, por lo que las ráfagas son más cortas que los CPU Bound

# Algoritmos de planificación – CPU + I/O

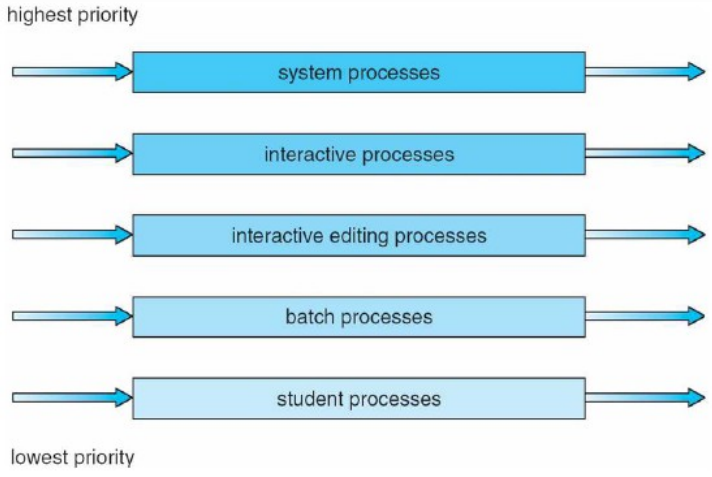
* Ciclo de vida de un proceso: Uso de CPU + Operaciones de I/O
* Cada dispositivo tiene su cola de procesos en espera y un planificador por cada cola
* Se considera I/O independiente de la CPU (DMA, PCI, etc.)
  + Uso de CPU y operaciones de I/O en simultaneo

# Algoritmos de planificación – Criterios de desempate

* Orden de aplicación
  + Orden de llegada de los procesos
  + PID de los procesos
* Se mantiene siempre la misma política

# Colas Multinivel

* Los planificadores actuales son combinaciones de los algoritmos anteriores
* La cola de listos es dividida en varias colas
* Los procesos se colocan en colas según una clasificación que realiza el sistema operativo
* Cada cola posee su propio algoritmo de planificación denominado “Planificador Horizontal”
* A su vez, existe un algoritmo que planifica las colas denominado “Planificador Vertical”
* Retroalimentación
  + Un proceso puede cambiar de una cola a la otra



# Planificación con múltiples procesadores

* La planificación es más compleja cuando hay múltiples CPUs
* La carga se divide entre distintas CPUs logrando capacidades de procesamiento mayores
* Si un procesador falla, el resto toma el control

## Criterios

* Planificación temporal
  + Qué proceso y durante cuánto tiempo
* Planificación espacial
  + En qué procesador ejecutar
  + Huella
    - Estado que el proceso va dejando en la caché de un procesador
  + Afinidad
    - Preferencia de un proceso para ejecutar en un procesador
* La asignación de procesos a un procesador puede ser
  + Estática
    - Existe una afinidad de un proceso a una CPU
  + Dinámica
    - La carga se comparte (balanceo de carga)
* La política puede ser
  + Tiempo compartido
    - Se puede considerar una cola global o una cola local a cada procesador
  + Espacio compartido
    - Grupos (threads / hilos)
    - Particiones

## Clasificaciones

* Procesadores homogéneos
  + Todas las CPUs son iguales
  + No existen ventajas físicas sobre el resto
* Procesadores heterogéneos
  + Cada procesador tiene su propia cola, su propio clock y su propio algoritmo de planificación

### Otra clasificación

* Procesadores débilmente acoplados
  + Cada CPU tiene su propia memoria principal y canales
* Procesadores fuertemente acoplados
  + Comparten memoria y canales
* Procesadores especializados
  + Uno o más procesadores principales de uso general y uno o más procesadores de uso específico

# Memoria

La parte del SO que administra la memoria se llama “Administrador de memoria”

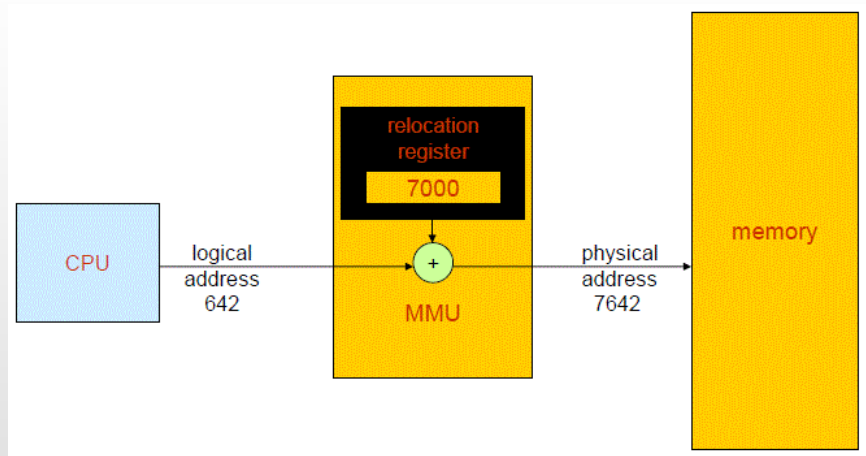
* Lleva un registro de las partes de la memoria que se están utilizando y de aquellas que no
* Asigna espacio en memoria a los procesos cuando estos la necesitan
* Libera espacio de memoria asignada a procesos que han terminado

Se espera que el SO haga uso eficiente de esta memoria con el fin de alojar el mayor número de procesos (multiprogramación)

## Direccionamiento

* Dirección Lógica
  + Es una dirección que enmascara o abstrae una dirección física
  + Referencia una localidad en memoria
  + Se la debe traducir a una dirección física
* Dirección Física
  + Es la dirección real con la que se accede efectivamente a memoria
  + Representa la dirección absoluta en memoria principal
* La CPU trabaja con direcciones lógicas y para acceder a la memoria se deben transformar en direcciones físicas
* El mapeo entre direcciones virtuales y físicas se realiza mediante el hardware (MMU)

Traducción MMU



## Asignación de memoria

* Particiones Fijas
  + La memoria se divide en particiones o regiones de tamaño fijo (tamaños iguales o diferentes)
  + Alojan un único proceso
  + Cada proceso se coloca en alguna partición de acuerdo a algún criterio
    - First Fit
    - Best Fit
    - Worst Fit
    - Next fit
  + Fragmentación Interna
* Particiones Dinámicas
  + Las particiones varían en tamaño y número
  + Alojan un proceso cada una
  + Cada partición se genera en forma dinámica del tamaño justo que necesita el proceso
  + Fragmentación Externa

## Fragmentación

Se produce cuando una localidad de memoria no puede ser utilizada por no encontrarse en forma contigua

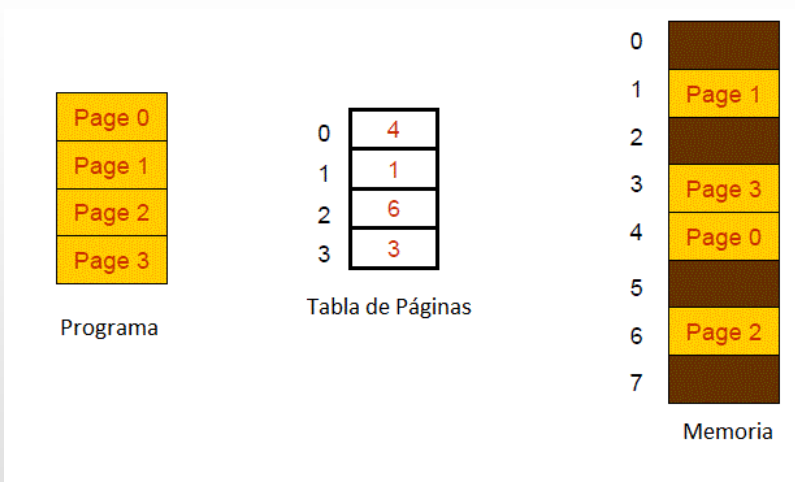
* Fragmentación Interna
  + Se produce en el esquema de particiones fijas, por ejemplo
  + Es interna a la localidad asignada
  + Es la porción de la localidad que queda sin utilizar
* Fragmentación Externa
  + Se produce en el esquema de particiones dinámicas, por ejemplo
  + Son huecos que van quedando en la memoria a medida que los procesos finalizan
  + Al no encontrarse en forma contigua puede darse el caso de que tengamos memoria libre para alocar un proceso, pero que no la podamos utilizar
  + Se soluciona con la compactación pero es muy costosa

## Paginación

* La memoria se divide en porciones de igual tamaño llamadas marcos
* El espacio de direcciones de los procesos se divide en porciones de igual tamaño denominadas páginas
* Tamaño páginas = Tamaño marco = 512 Bytes
* El SO mantiene una tabla de páginas para cada proceso, la cual contiene el marco donde se encuentra cada página
* La paginación bajo demanda es un técnica eficiente de manejar esta estrategia (Puede causar trashing)
* Puede causar fragmentación interna

Direccionamiento con paginación

* Un proceso en ejecución hace referencia a una dirección virtual
* El SO busca la página en la tabla de páginas del proceso y determina en qué marco se encuentra
* La dirección física se forma por la concatenación de la dirección de inicio del marco que aloja la página y un desplazamiento



## Segmentación

* Se puede ver como una mejora de la paginación
* Puede haber fragmentación externa
* Ahora la tabla de segmentos, además de tener la dirección de inicio del mismo, tiene la longitud o límite
* Las direcciones lógicas constan de dos partes
  + Un número de segmento
  + Un desplazamiento dentro del segmento

## Memoria virtual con paginación

* La técnica de paginación intenta alocar la mayor cantidad de paginas necesarias posibles
* Cada vez que hay que alocar una pagina en un marco, se produce un fallo de página (hard page fault)
* Si no hay espacio necesario para alocar una página, hay que seleccionar una página víctima, con lo cual existen varios algoritmos
* La mayoría de los algoritmos predicen el comportamiento futuro mirando el comportamiento pasado

### Algoritmo optimo

* Selecciona la pagina cuya próxima referencia se encuentra mas lejana a la actual
* Imposible de implementar

### Least Recently Used

* Reemplaza la página que no fue referenciada por más tiempo
* Cada página debe tener información del instante de su ultima referencia, lo que causa mayor overhead

### First In First Out

* Trata a los frames en uso como una cola circular
* Simple de implementar
* La pagina mas vieja en la memoria es reemplazada
* La pagina puede ser necesitada pronto

### First In First Out con Segunda chance

* Se utiliza un bit adicional de referencia
* Cuando la pagina se carga en memoria, el bit R se pone en 0
* Cuando la pagina es referenciada el bit R se pone en 1
* La victima se busca en orden fifo, se selecciona la primer pagina cuyo bit está en 0
* Mientras se busca la victima, cada bit R que tiene el valor 1 se cambia a 0 y se reencola

## Asignación de marcos en paginación

* Asignacion fija
  + A cada proceso se le asigna una cantidad arbitraria de marcos
  + A su vez, para el reparto se puede usar
    - Reparto equitativo
      * Se asigna la misma cantidad de marcos a cada proceso
    - Reparto proporcional
      * Se asignan marcos en base a la necesidad que tiene cada proceso
* Asignacion dinámica
  + Los procesos se van cargando en forma dinámica de acuerdo a la cantidad de marcos que necesiten

## Alcance del reemplazo

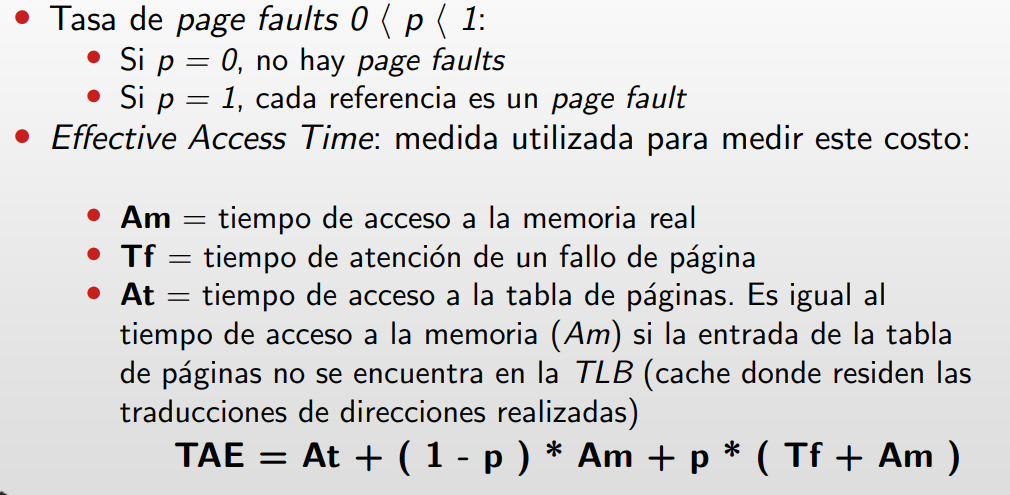
* Reemplazo Global
  + El fallo de página de un proceso puede reemplazar la página de cualquier proceso
* Reemplazo Local
  + El fallo de página de un proceso solo puede reemplazar sus propias paginas

## Descarga asincronica de paginas

* El sistema operativo reserva uno o varios marcos para la descarga asincrónica de paginas
* Cuando es necesario descargar una pagina modificada
  + La pagina que provoco el fallo se coloca en un frame designado a la descarga asincrónica
  + El SO envía la orden de descargar asincrónicamente la pagina modificada mientras continua la ejecución de otro proceso
  + El frame de descarga asincrónica pasa a ser el que contenia a la pagina victima que ya se descargó correctamente

## Performance

* La técnica de paginación por demanda puede generar una degradación de rendimiento del sistema debido a que el reemplazo de paginas es costoso



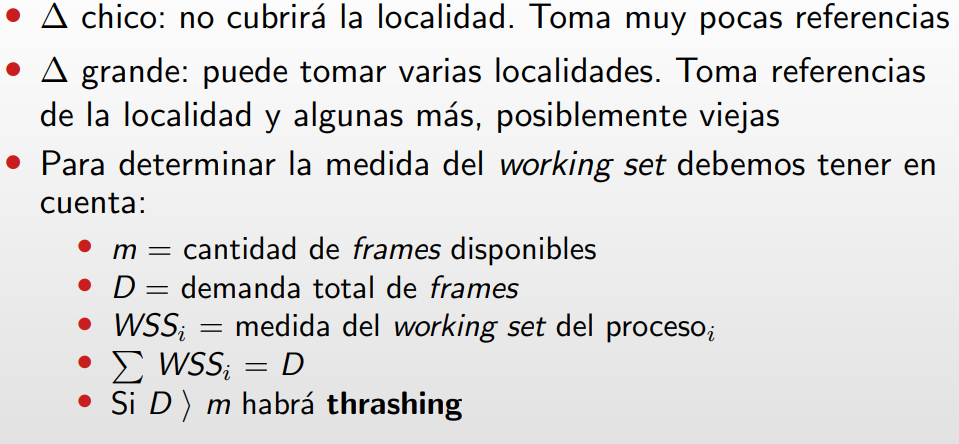
## Trashing

* Se dice que un sistema está en trashing o hiperpaginación cuando pasa más tiempo paginando que ejecutando procesos
* Si un proceso cuenta con todos los frames que necesita, no habría trashing. Salvo excepciones como la anomalía de Belady
* Existen técnicas para evitarlo
  + Estrategia de working set

### Modelo de Localidad

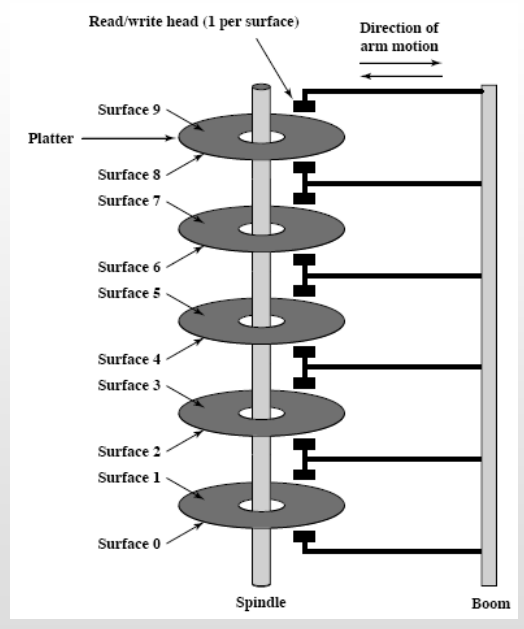
* Las referencias a datos y programas dentro de un proceso tienden a agruparse
* La localidad de un proceso en un momento dado se da por el conjunto de paginas que son referenciadas en ese momento
* En cortos periodos de tiempo, el proceso necesitará pocas “piezas” del proceso (una página de instrucciones y otra de datos)
* Se define una ventana de trabajo (∆) que contiene las referencias de memoria más recientes
* Working set es el conjunto de páginas que tienen las ∆ referencias a páginas más recientes

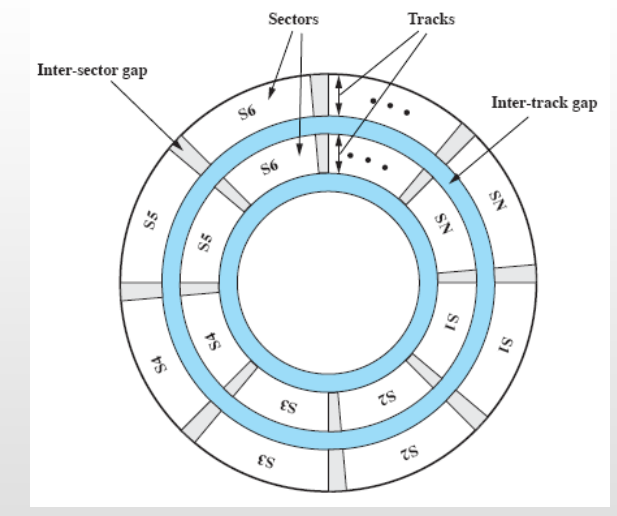
## Selección del ∆



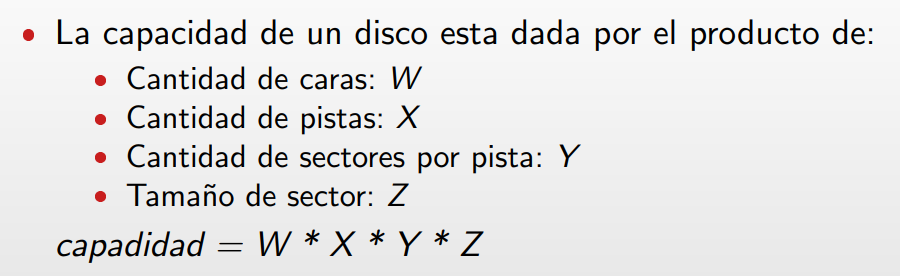
# Discos

## Organización física de un HDD

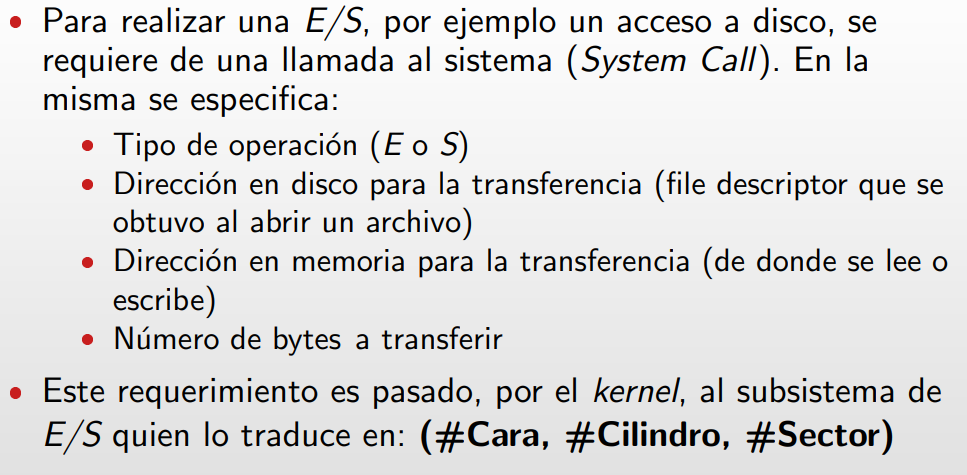




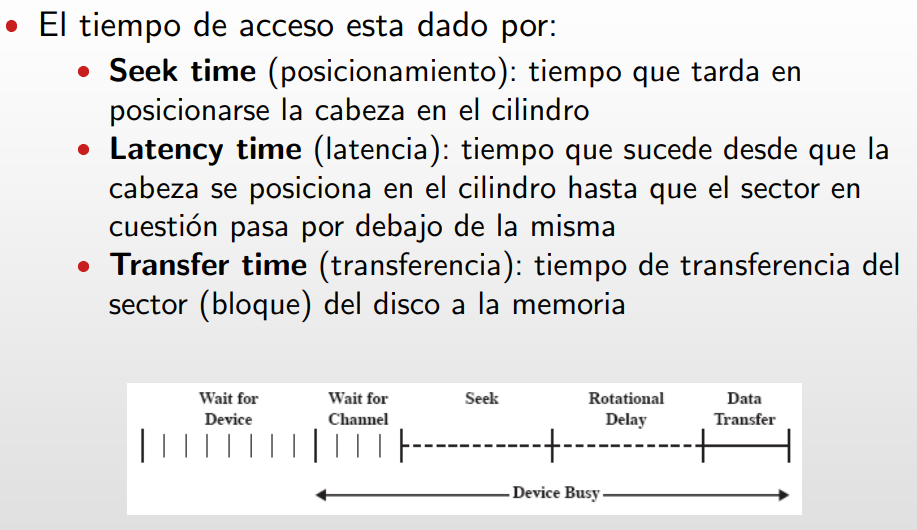
## Capacidad de un HDD



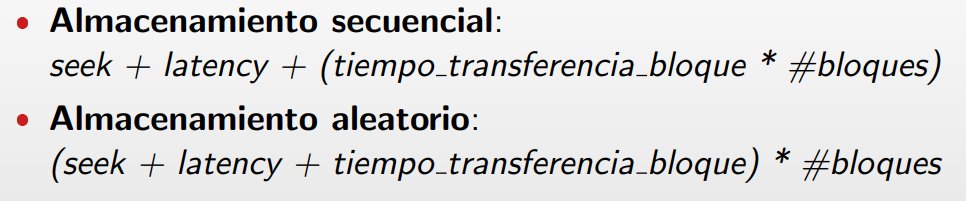
## Acceso a un HDD



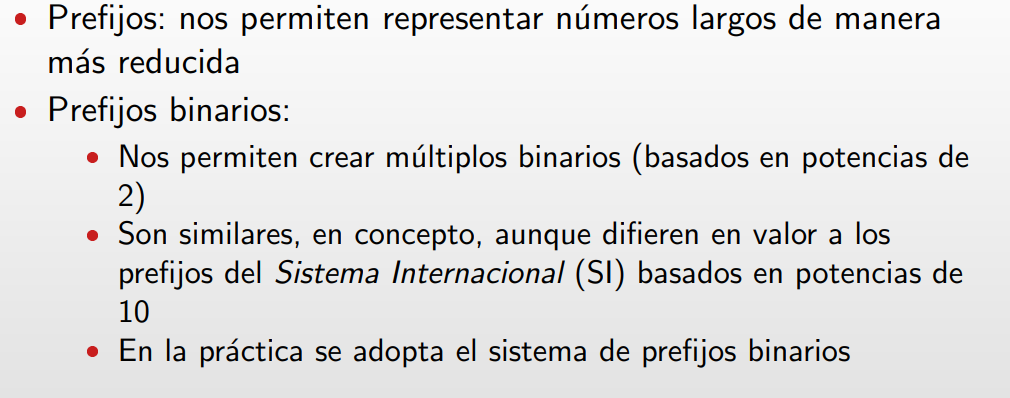
## Tiempo de acceso a un HDD



Si el tiempo de latencia no se conoce, se considera que es igual a lo que tarda el disco en dar media vuelta (tiempo de latencia promedio)

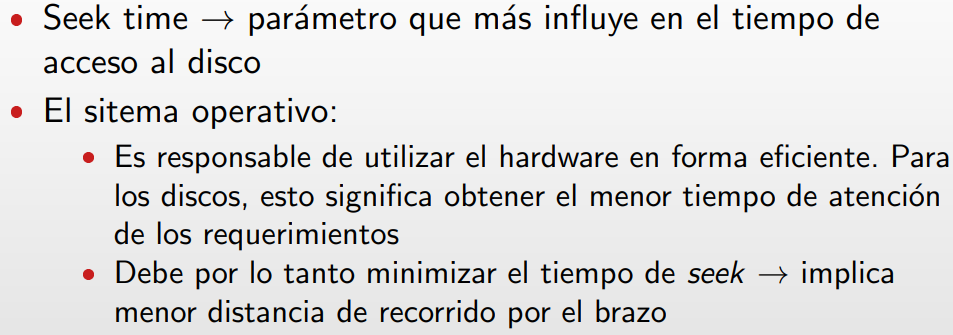


## Prefijos

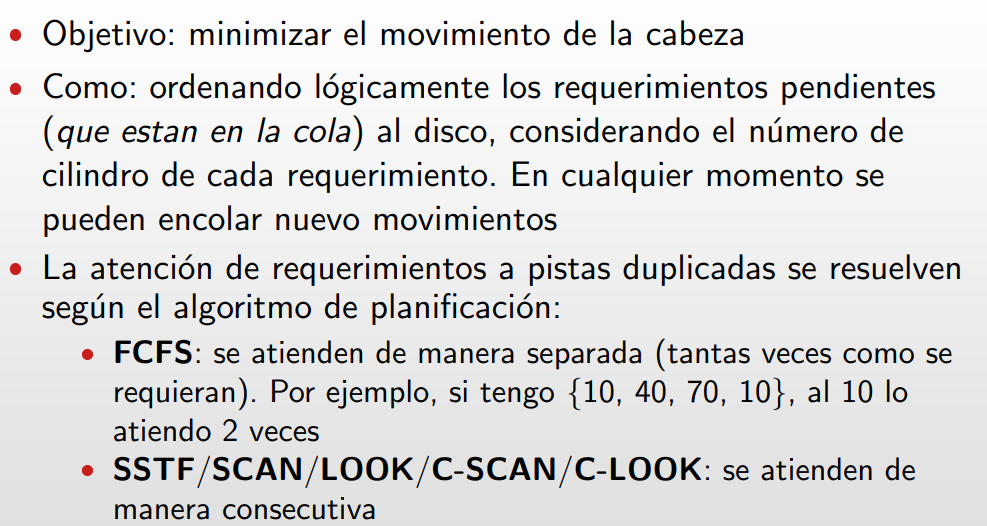


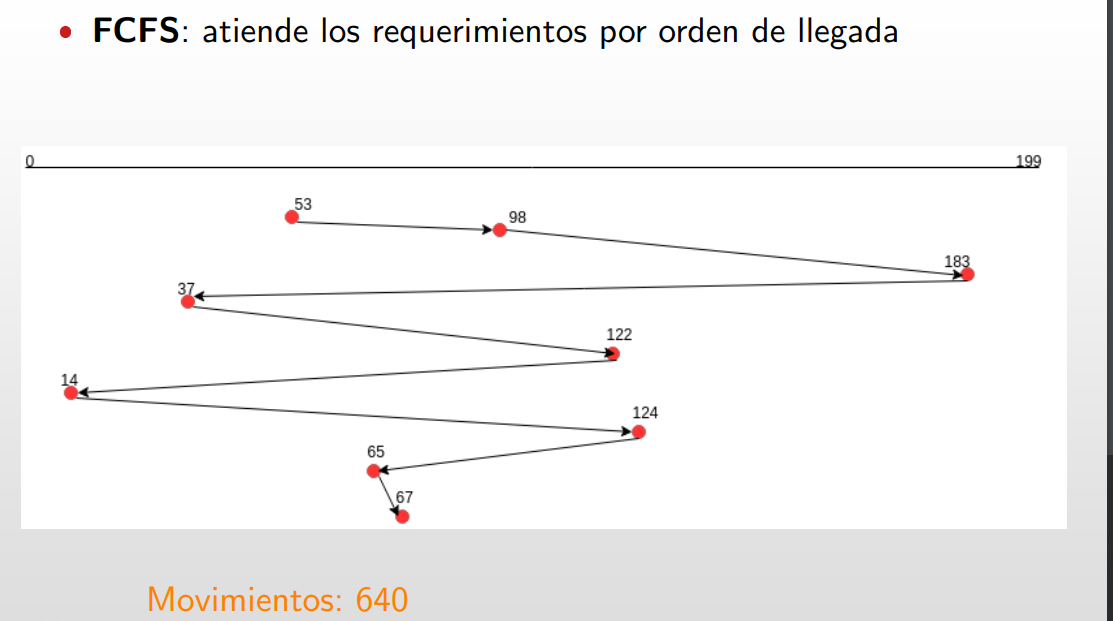


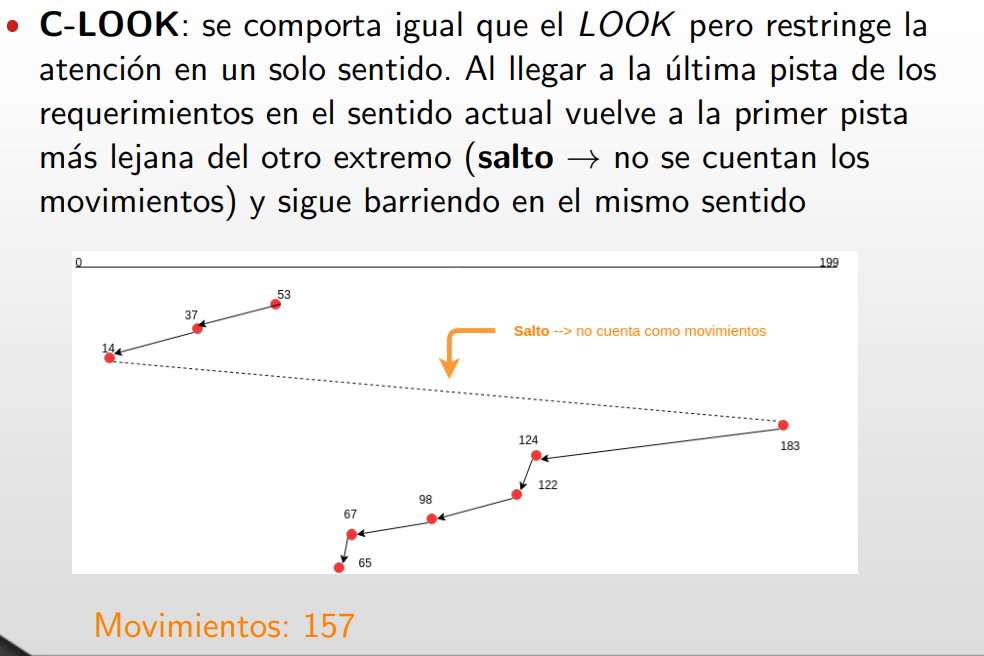
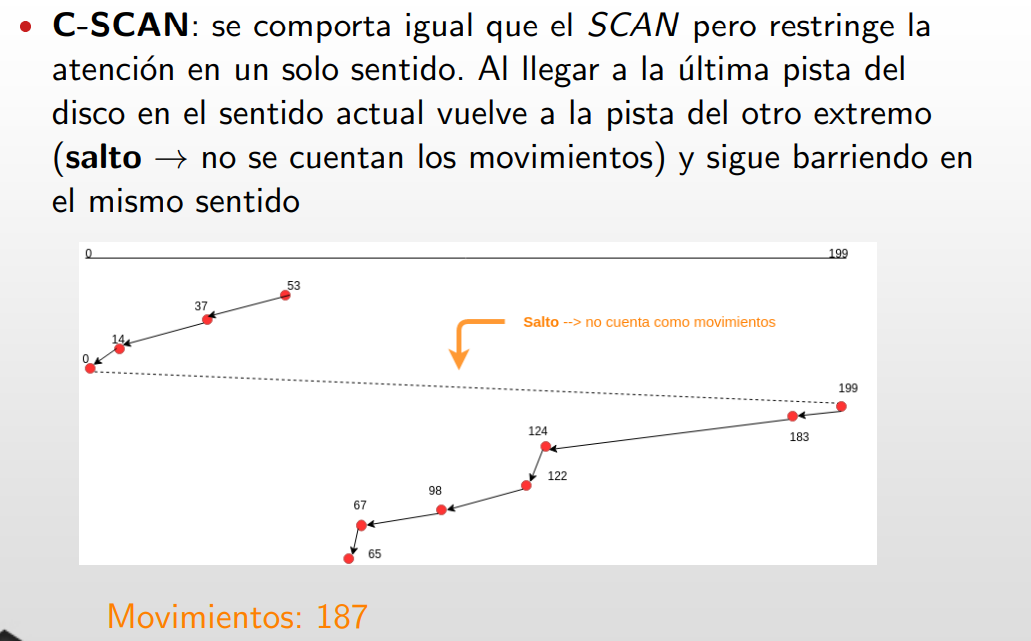
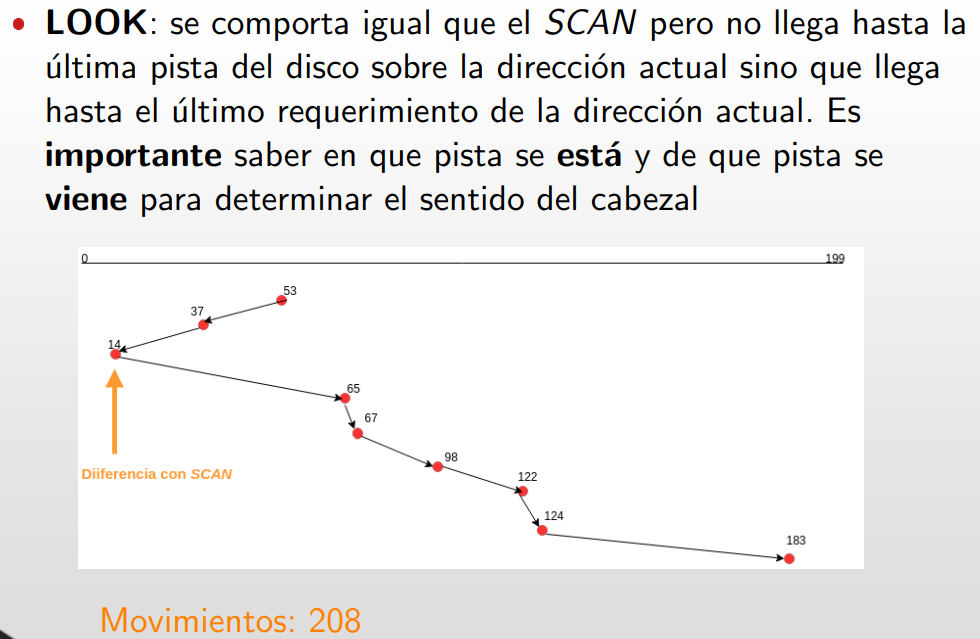
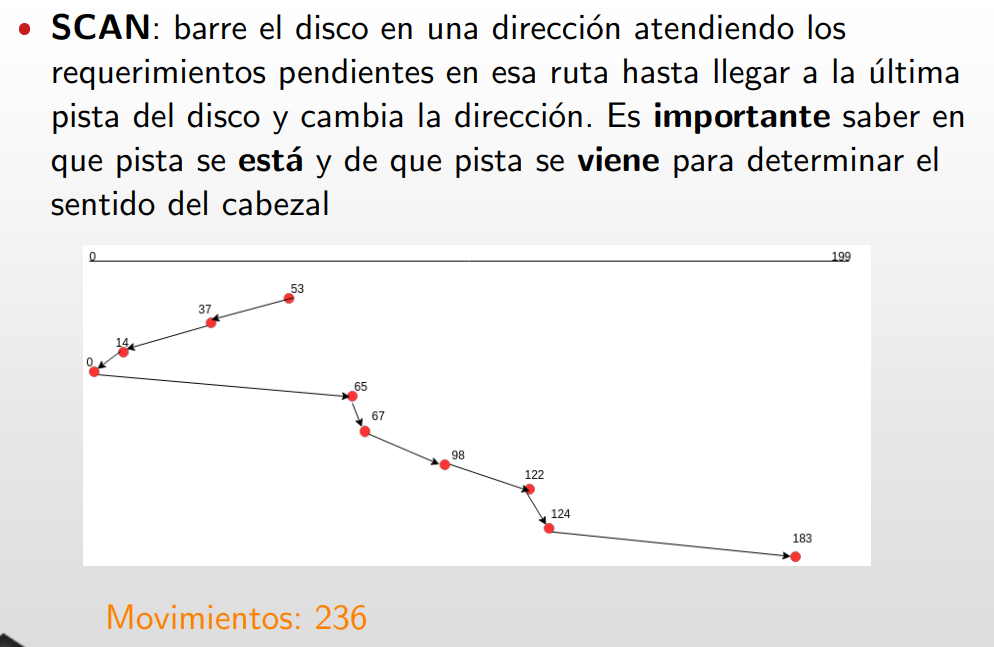
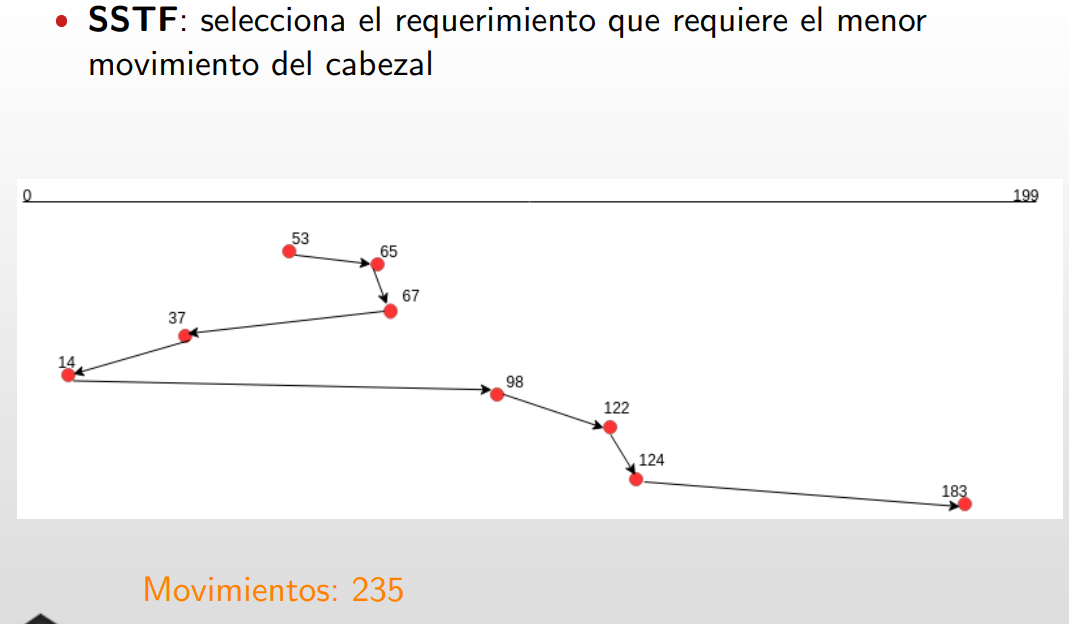
## Planificación de requerimientos de un HDD



## Algoritmos de planificación







## Atencion de page faults

