**Resumen PN – ISO**

**P4 – Procesos**

**Algoritmos**

FCFS: Cuando hay que elegir un proceso para ejecutar, se selecciona el más viejo. No favorece a ningún tipo de procesos, pero en principio podríamos decir que los CPU Bound terminan al comenzar su primera ráfaga, mientras que los I/O Bound no.

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

RR: Política basada en un reloj. Cuando un proceso es expulsado de la CPU es colocado al final de la RQ y se selecciona otro (FIFO circular). Existe un contador que indica las unidades de CPU en las que el proceso se ejecutó. Cuando el mismo llega a 0, el proceso es expulsado. El contador puede ser global o local (PCB).

* TV: El contador se inicializa en Q cada vez que un proceso es asignado a la CPU. Es el más utilizado.
* TF: El contador se inicializa en Q cuando su valor es 0. Se puede ver como un valor de Q compartido entre los procesos.

STRF: Version preemptive de SJF. Selecciona el proceso al cual le resta menos tiempo de ejecución en su siguiente ráfaga. Favorece a procesos I/O Bound.

Prioridades: Cada proceso tiene un valor que representa su prioridad. Se selecciona el proceso de mayor prioridad de los que se encuentran en la RQ. Existe una RQ por cada nivel de prioridad. Puede ser apropiativo o no. Los procesos de baja prioridad pueden sufrir inanición. Solución: permitir a un proceso cambiar su prioridad durante su ciclo de vida (Aging o Penalty).

Tiempos:

* Retorno: tiempo que transcurre entre que el proceso llega al sistema hasta que completa su ejecución (fin – (inicio+1)).
* Espera: tiempo que el proceso se encuentra en el sistema esperando, es decir el tiempo que pasa sin ejecutarse (retorno – CPU).

Criterios de desempate:

* Orden de llegada de los procesos.
* PID de los procesos.

**P5 – Memoria**

**Traducción de direcciones**

Dirección lógica a física:

N pag = dir lógica DIV tam pag

Desplazamiento = dir lógica MOD tam pag

Inicio marco = inicio marco en el que está N pag

Dir física = inicio marco + desplazamiento

Dirección física a lógica:

N marco = dir física DIV tam marco

Desplazamiento = dir física MOD tam marco

N pag = N pag que esté en N marco

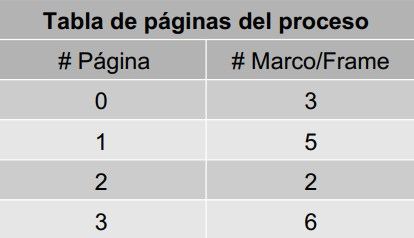
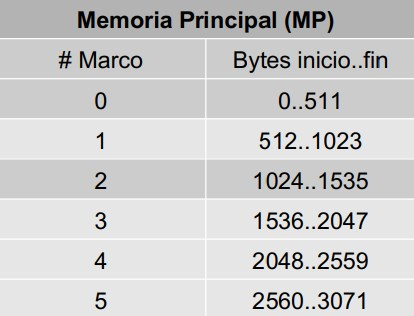
Dir lógica = (N pag \* tam pag) + desplazamiento

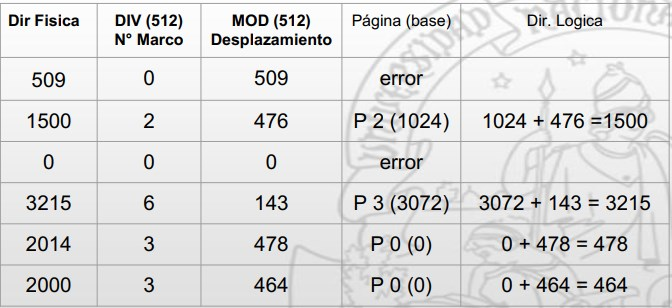
Ejemplo:

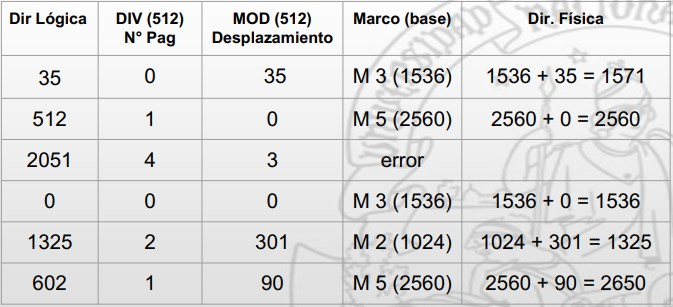
Tamaño página = 512 bytes

Tamaño dirección = 1 byte

Tamaño proceso = 2000 bytes







**Direcciones**

Ejemplo:

* Se dispone de un espacio de direcciones virtuales de 32 bits.
* Cada dirección referencia 1 byte.
* El tamaño de página es de 512 KiB.
* Se dispone de 256 MiB de memoria real.
* El tamaño de cada entrada en la tabla de páginas es 2 Kb.

Cantidad de direcciones:

232 = 4.294.967.296

Tamaño máximo de un proceso:

cant direcciones \* tam referencia = 232 \* 1 byte = 232 bytes = 4GiB

Número máximo de páginas que puede tener un proceso:

tam max proceso / tam pag = 4194304 KiB / 512 KiB = 8192 páginas

Número de marcos que puede haber:

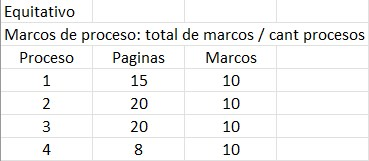
memoria / tam pag = 262144 KiB / 512 KiB = 512 frames

Tamaño máximo de la tabla de cada proceso:

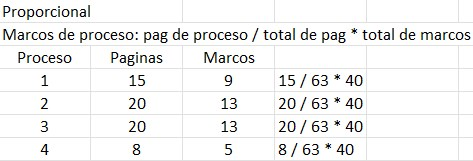
cant max pag \* tam entrada pag = 8192 \* 2 Kb = 16384 Kb

**Reparto de marcos (asignación fija)**

Marcos = 40

Reparto equitativo:

Reparto proporcional:



**Algoritmos**

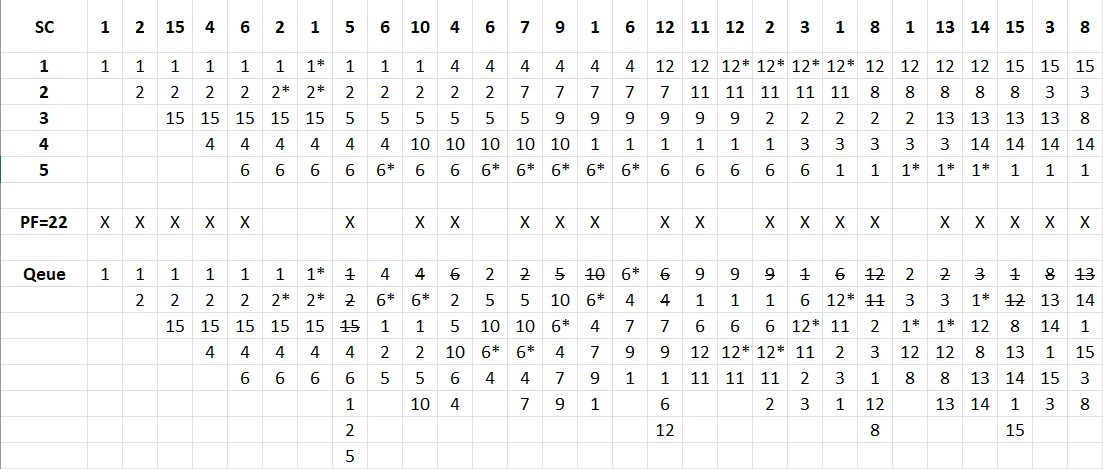
Óptimo: Selecciona la página cuya próxima referencia se encuentra más lejana a la actual.

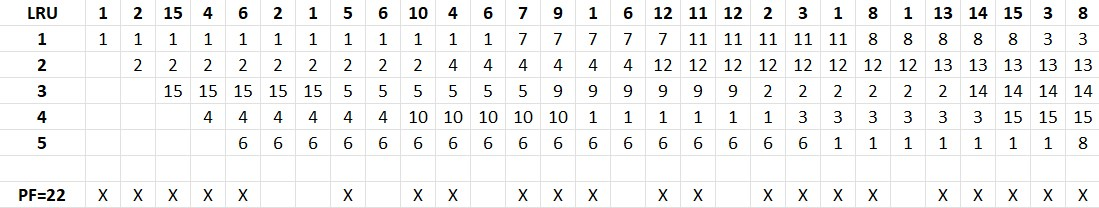
FIFO: La página más vieja en la memoria es reemplazada.

FIFO segunda chance: Se usa el bit de referencia. Cuando la página se carga en memoria, el bit R se pone a 0. Cuando la página es referenciada el bit R se pone en 1. La víctima se busca en orden FIFO. Se selecciona la primera página cuyo bit R esté en 0. Mientras se busca la víctima cada bit R que tiene el valor 1 de las que podrían haber sido seleccionadas (no todas las que tengan el bit en 1) se cambia a 0.

LRU: Reemplaza la página que no fue referenciada por más tiempo.

Ejemplo: P5EJ22

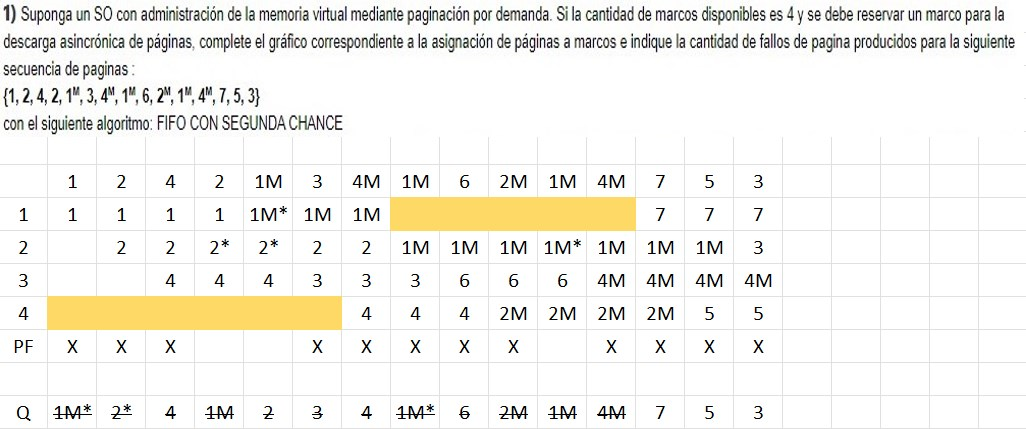
FIFO segunda chance:

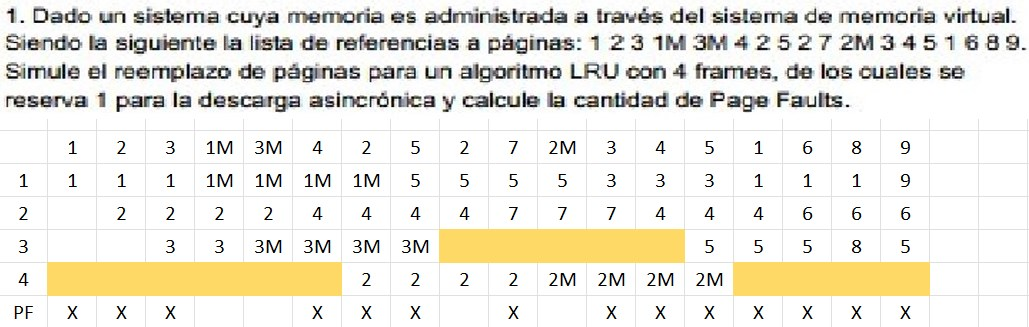
LRU:

Descarga asincrónica

El sistema operativo reserva uno o varios marcos para la descarga asincrónica de páginas. Cuando es necesario descargar una página modificada:

* La página que provocó el fallo se coloca en un frame designado a la descarga asincrónica.
* El SO envía la orden de descargar asincrónicamente la página modificada mientras continúa la ejecución de otro proceso.
* El frame de descarga asincrónica pasa a ser el que contenía a la página víctima que ya se descargó correctamente.

Ejemplos (no están corregidos, pero creo que están bien):

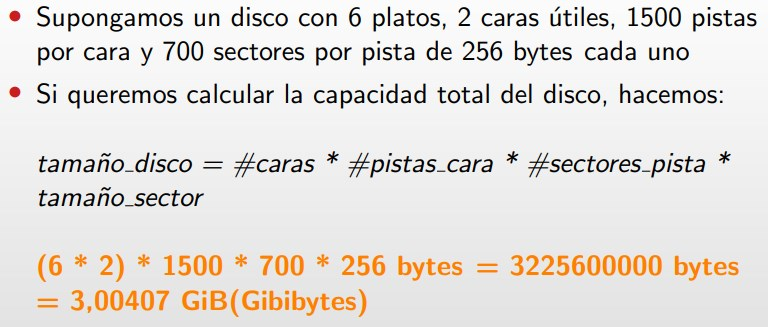


**P6 – Entrada Salida**

**HDD**

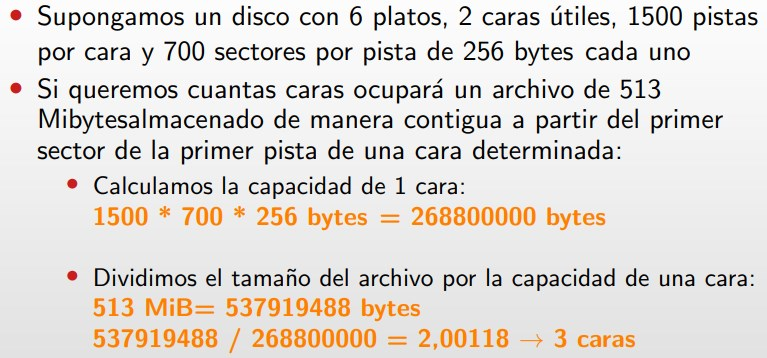
Capacidad:

platos \* caras por plato \* pistas (cilindros) por cara \* sectores por pista \* tam sector

Ejemplo:

Ocupación:

Ejemplo:



Tiempos:

* Seek time (posicionamiento): tiempo que tarda en posicionarse la cabeza en el cilindro.
* Latency time (latencia): tiempo que sucede desde que la cabeza se posiciona en el cilindro hasta que el sector en cuestión pasa por debajo de la misma. Si no se conoce, se asume media vuelta.
* Transfer time (transferencia): tiempo de transferencia del sector (bloque) del disco a la memoria.

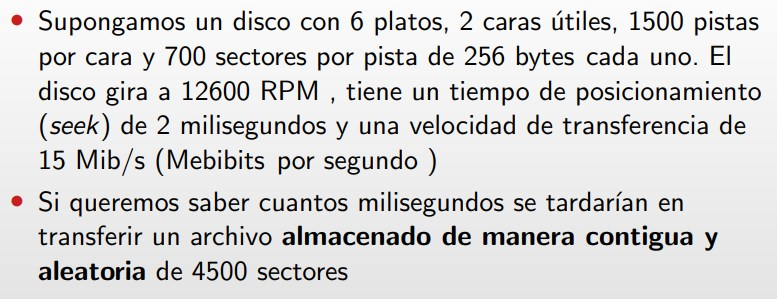
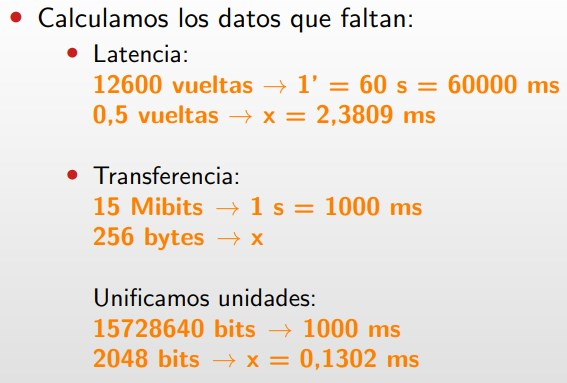
Tiempo de acceso:

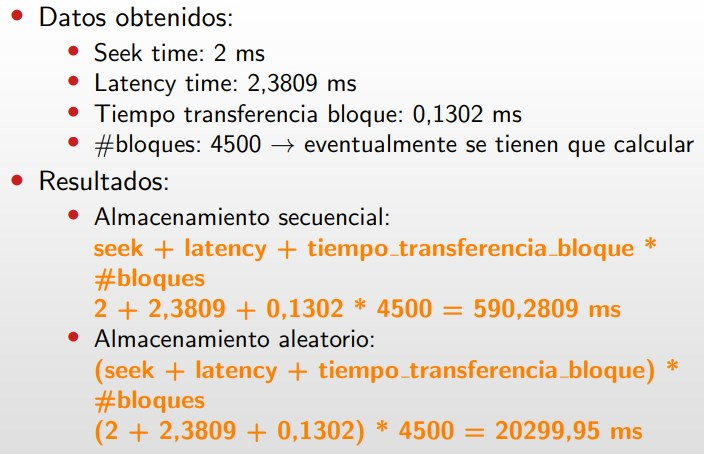
* Secuencial:

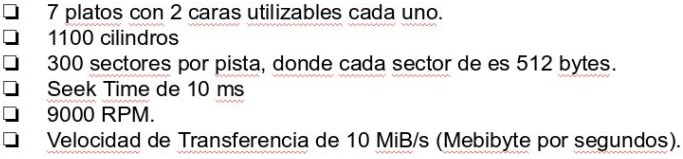
seek + latency + (transfer bloque \* cant bloques)

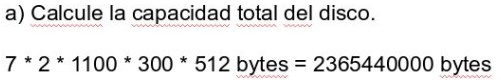
* Aleatorio:

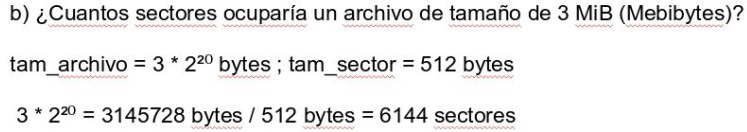
(seek + latency + transfer bloque) \* cant bloques

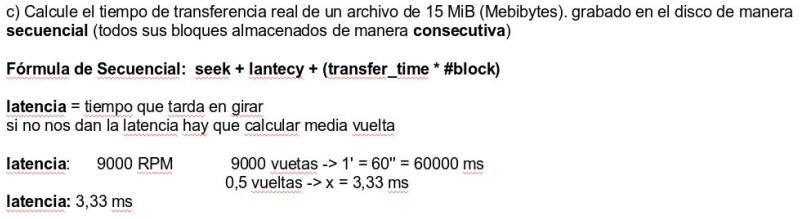
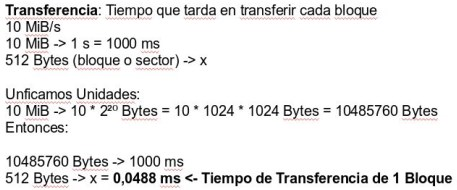
Ejemplo:

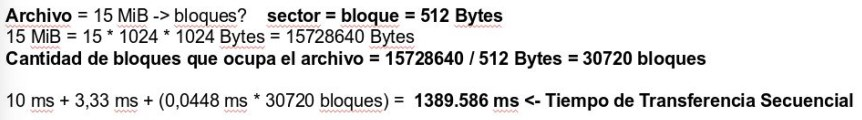


Ejemplo:







****

**Algoritmos**

El seek time es el parámetro que más influye en el tiempo de acceso al disco. El objetivo es minimizar el movimiento de la cabeza.

Ejemplo:

Cantidad de pistas: 200.

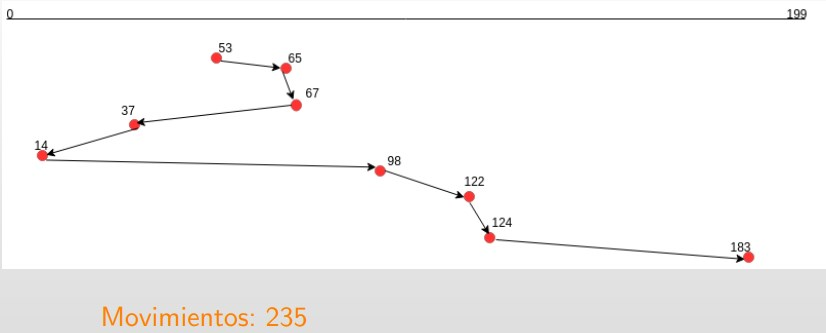
Requerimientos: {98, 183, 37, 122, 14, 124, 65, 67}

Viene de: 61.

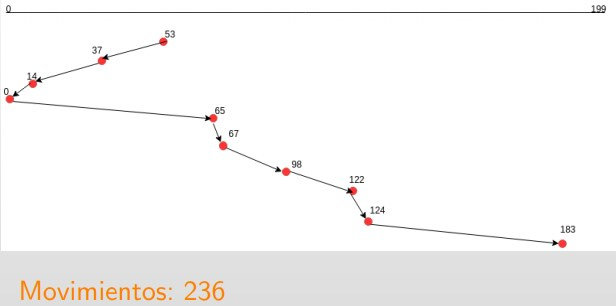
Ubicación actual: 53.

FCFS: atiende los requerimientos por orden de llegada.

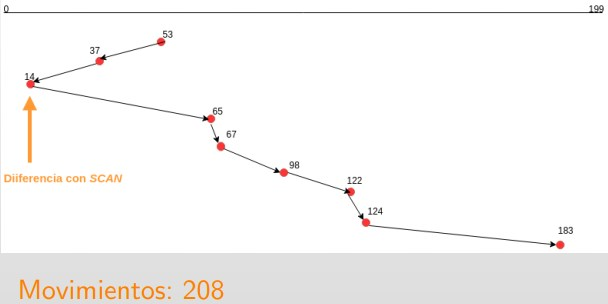
SSTF: selecciona el requerimiento que requiere el menor movimiento del cabezal.

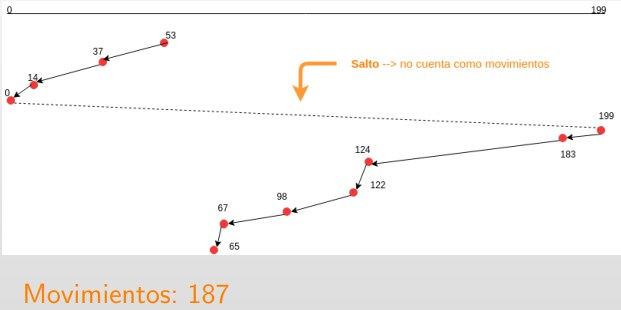


SCAN: barre el disco en una dirección atendiendo los requerimientos pendientes en esa ruta hasta llegar a la última pista del disco y cambia la dirección. Es importante saber en qué pista se está y de que pista se viene para determinar el sentido del cabezal.

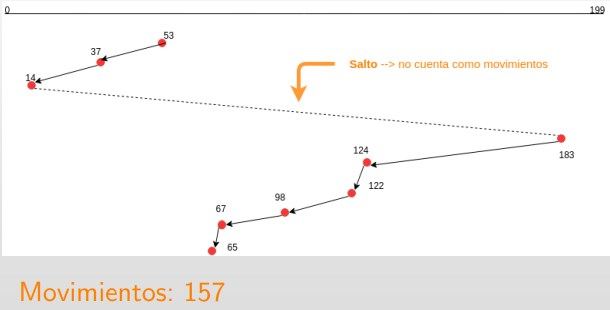


LOOK: se comporta igual que el SCAN, pero no llega hasta la última pista del disco sobre la dirección actual, sino que llega hasta el último requerimiento de la dirección actual. Es importante saber en qué pista se está y de que pista se viene para determinar el sentido del cabezal.



C-SCAN: se comporta igual que el SCAN, pero restringe la atención en un solo sentido. Al llegar a la última pista del disco en el sentido actual, vuelve a la pista del otro extremo (salto -> no se cuentan los movimientos) y sigue barriendo en el mismo sentido.

C-LOOK: se comporta igual que el LOOK, pero restringe la atención en un solo sentido. Al llegar a la última pista de los requerimientos en el sentido actual, vuelve a la primera pista más lejana del otro extremo (salto -> no se cuentan los movimientos) y sigue barriendo en el mismo sentido.



Existen requerimientos especiales que deben atenderse con urgencia. Los fallos de página indican que tienen mayor prioridad con respecto a los requerimientos convencionales, por lo tanto, deben ser atendidos inmediatamente después del requerimiento que se está atendiendo actualmente. La lógica de atención de múltiples PF se maneja según el algoritmo de planificación. En todos los algoritmos, los movimientos utilizados para atender estos requerimientos especiales deben ser contados.

Ejemplo: {10, 40PF, 70PF, 10}

* FCFS: primero se atiende al 40PF y luego al 70PF.
* SSTF: si estoy en la pista 65, primero atiendo al 70PF y luego al 40PF.

Una vez que no existan más requerimientos por PF en la cola, se procede:

* FCFS: en orden FCFS.
* SSTF: en orden SSTF.
* SCAN: con el sentido que determina la atención de los últimos dos requerimientos. Puede cambiar de sentido.
* C-SCAN: con el sentido original.
* LOOK: del mismo modo en que lo hace el SCAN.
* C-LOOK: del mismo modo en que lo hace el C-SCAN.

Ejemplo:

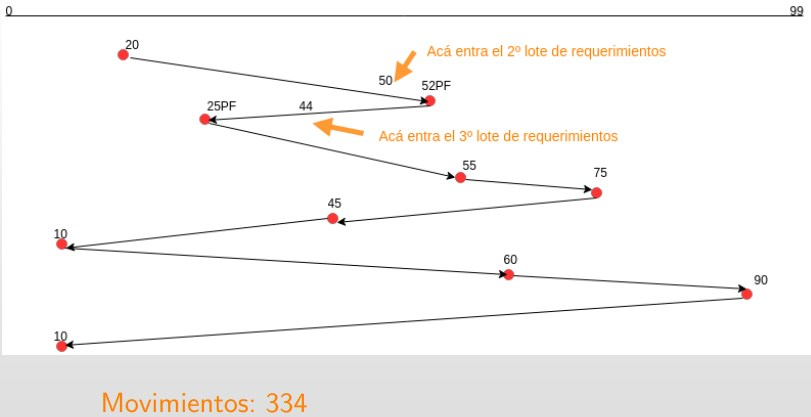
Cantidad de pistas: 100.

Requerimientos: {55, 75, 52PF, 45, 10}. Luego de 30 movimientos {25PF, 60}. Luego de 10 movimientos más {90, 10}.

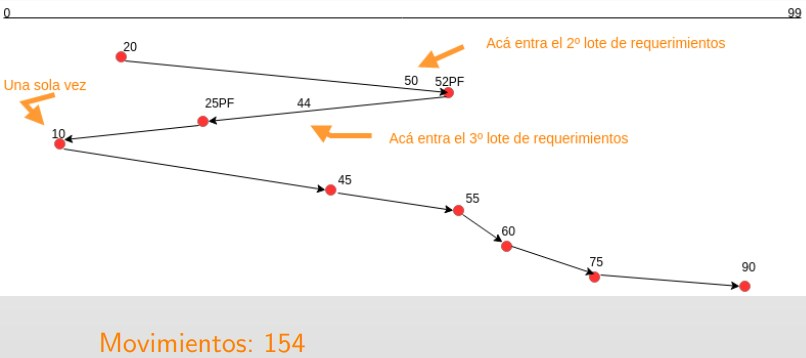
Viene de: 15.

Ubicación actual: 20.

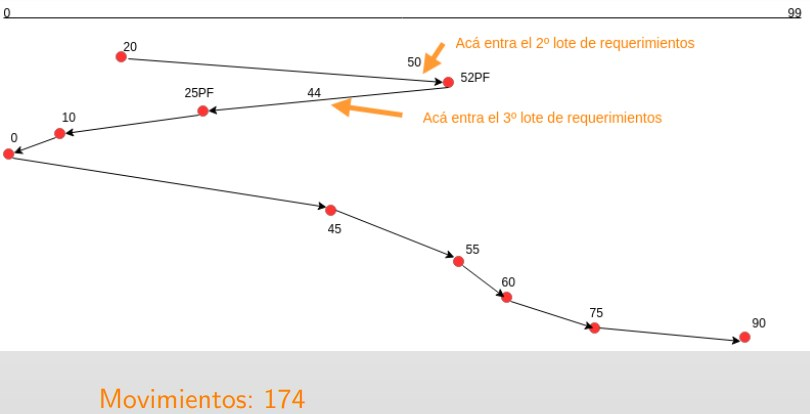
FCFS:



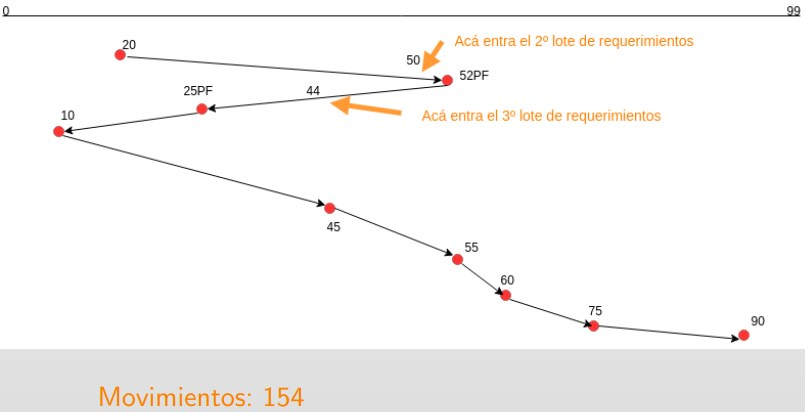
SSTF:



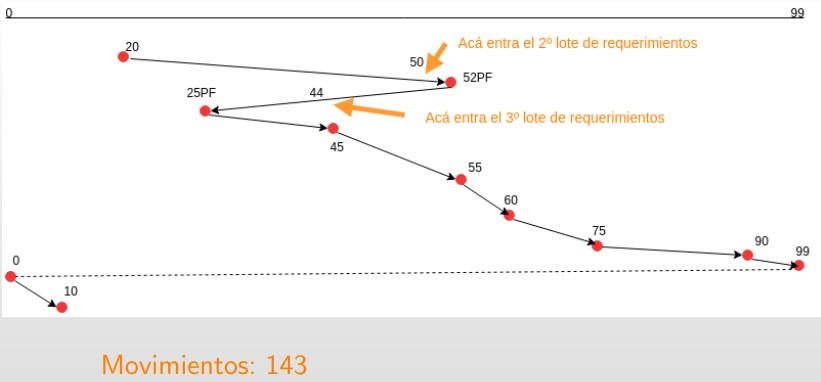
SCAN:



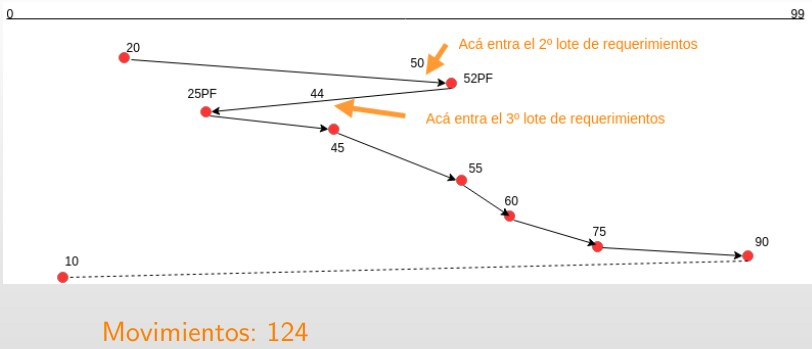
LOOK:



C-SCAN:



C-LOOK:

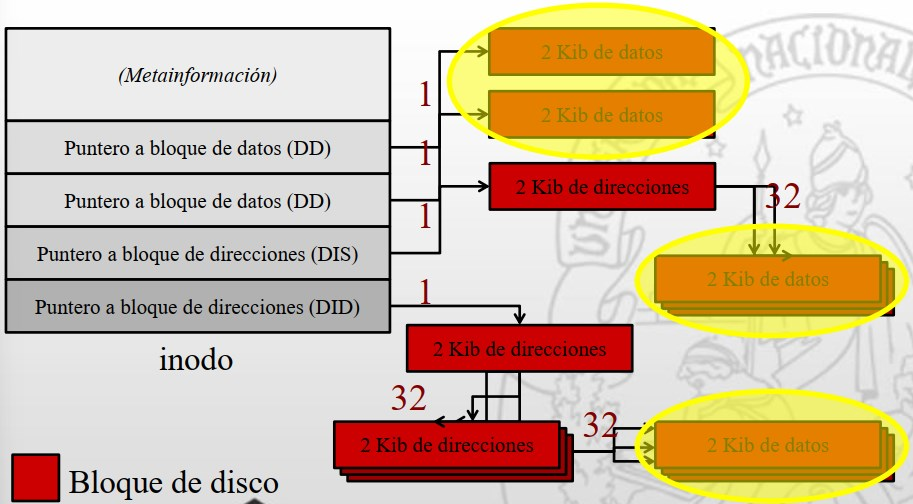


**Inodos**

Inodo: Estructura auxiliar que permite referenciar los archivos y acceder a ellos. Contiene metainformación del archivo y punteros a los bloques de datos en el disco que conforman el archivo. Se identifican con un número.

Estructura imaginaria (cada dirección es de 64 bits):

4 direcciones a los bloques de datos:

* 2 de direccionamiento directo (DD).
* 1 de direccionamiento indirecto simple (DIS).
* 1 de direccionamiento indirecto doble (DID).

Tamaño del inodo:

tam dirección \* direcciones = 64 bits \* 4 = 256 bits

Cantidad de inodos que puede tener un bloque de disco (de 2 Kib):

tam bloque / tam inodo = 2 Kib / 256 bits = 8 inodos

Direcciones por bloque de direcciones:

tam bloque / tam dirección = 2 Kib / 64 bits = 32 direcciones

Tamaño máximo de archivo:

2 punteros a bloques de datos (DD) +

1 puntero a bloque de direcciones que apuntan a bloques de datos (DIS) +

1 puntero a bloque de direcciones que apuntan a bloques de direcciones que apuntan a bloques de datos (DID) =

2 \* 2 Kib (DD) +

1 \* 32 \* 2 Kib (DIS) +

1 \* 32 \* 32 \* 2 Kib (DID) =

4 Kib + 64 Kib + 2048 Kib = 2116 Kib = 264.5 KiB