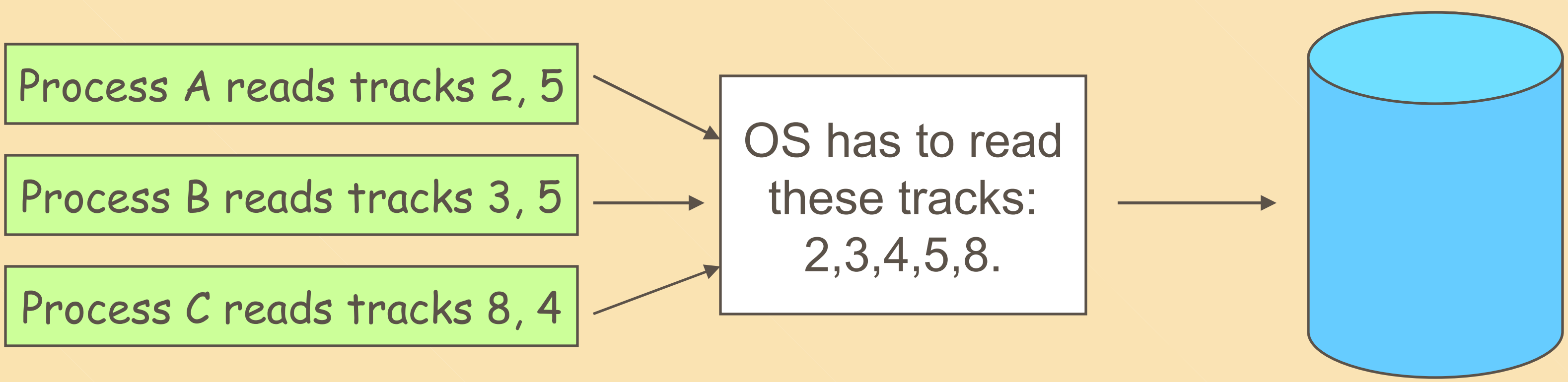
Disk Scheduling

Disk Scheduling

- At runtime, I/O requests for disk tracks come from the processes
- OS has to choose an order to serve the requests



Access time

- Total access time = seek time + rotational delay + data transfer time
- Seek time – time required to move the disk arm to the required track
- Rotational delay – time required to rotate the disk to the required sector
- Data transfer time – time to read/write data from/to the disk

Disk Scheduling

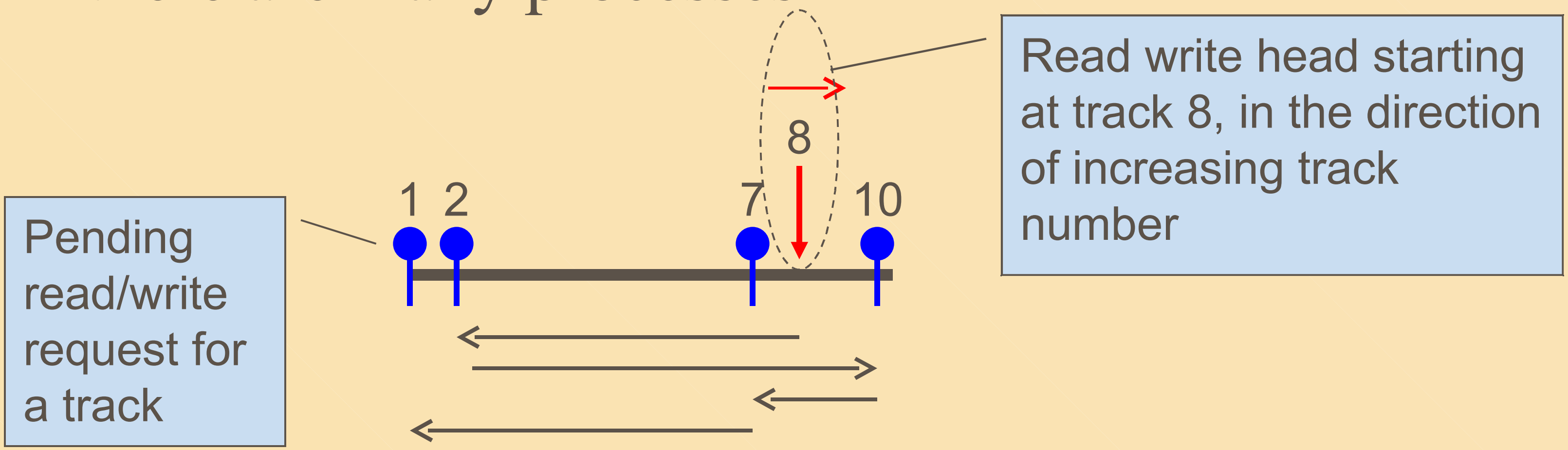
- The order that the read/write head is moved to satisfy several I/O requests
  - determines the total seek time
  - affects performance
  - the OS cannot change the rotational delay or transfer time, but it can try to find a ‘good’ order that spends less time in seek time.
- If requests are selected randomly, we will get the worst possible performance...

Disk Scheduling Policy

- FIFO: fair, but near random scheduling
- SSTF: possible starvation
- SCAN: favor requests for tracks near the ends
- C-SCAN
- FSCAN: avoid "arm stickiness" in SSTF, SCAN and C-SCAN

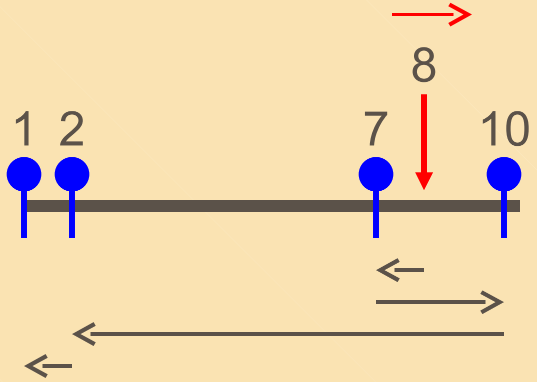
First-in-first-out, FIFO

- process requests in the order that the requests are made
- fair to all processes
- approaches random scheduling in performance if there are many processes



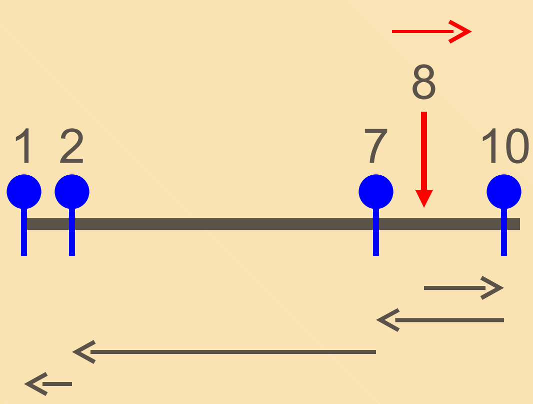
Shortest Service Time First, SSTF

- select the disk I/O request that requires the least movement of the disk arm from its current position
- always choose the minimum seek time
- new requests may be chosen before an existing request



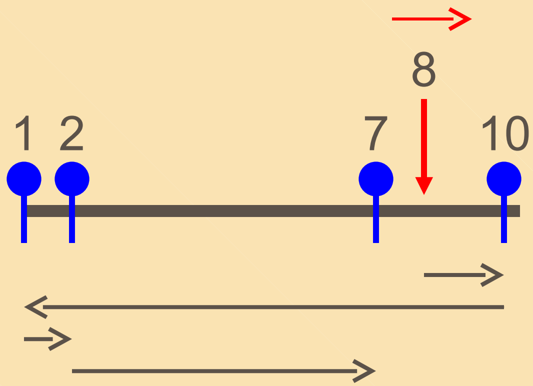
SCAN

- arm moves in one direction only, satisfying all outstanding requests until there is no more requests in that direction. The service direction is then reversed.
- favor requests for tracks near the ends



C-SCAN

- restrict scanning to one direction only
- when the last track has been visited in one direction, the arm is returned to the opposite end of the disk and the scan begins again.



FSCAN

- "Arm stickiness" in SSTF, SCAN, C-SCAN in case of repetitive requests to one track
- FSCAN uses two queues. When a SCAN begins, all of the requests are in one of the queues, with the other empty. During the scan, all new requests are put into the other queue.
- Service of new requests is deferred until all of the old requests have been processed.

Example

Trace the policies FIFO, SSTF, SCAN, C-SCAN and FSCAN for the following disk requests. Each I/O request on a track takes 5 time units. At time 0, the disk starts reading track 10, and the read/write head was moving to the larger track number direction .

Time	0	1	2	3	6	7
Request to access track ..	10	19	3	14	12	9

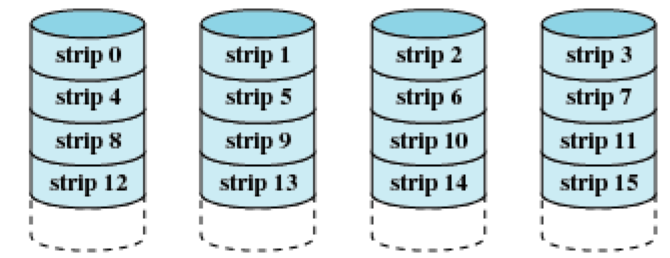
	Track access order	Average seek length
FIFO	10,19,3,14,12,9	(9+16+11+2+3)/5 = 8.2
SSTF	10,14,12,9,3,19	(4+2+3+6+16)/5 = 6.2
SCAN	10,14,19,12,9,3	(4+5+7+3+6)/5 = 5
C-SCAN	10,14,19,3,9,12	(4+5+16+6+3)/5 = 6.8
FSCAN	10,14,19,3,9,12	(4+5+16+6+3)/5 = 6.8

RAID

- Redundant Array of Independent Disks
- A set of physical disk drives viewed by the OS as a single logical drive
- Data are distributed across the physical drives. May improve performance.
- Redundant disk stores parity information. Recoverability, reliability.

RAID 0 (Non-redundant)

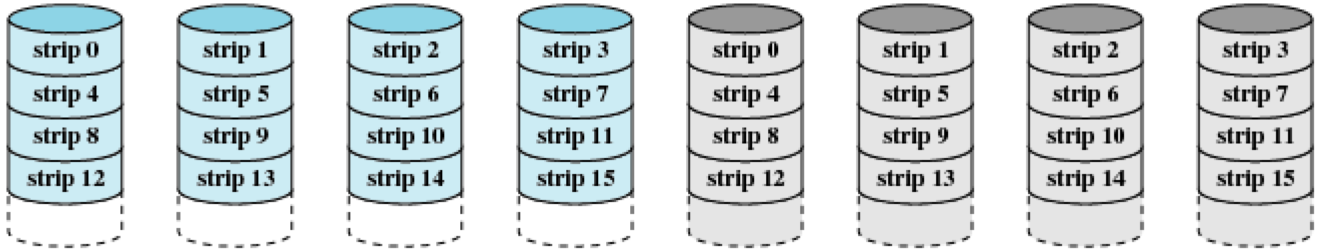
- The logical disk is divided into strips, mapped round robin to consecutive physical disks
- Improve performance in disk read/write
- Not fault tolerant



(a) RAID 0 (non-redundant)

RAID 1 (Mirrored)

- Each disk is mirrored by another disk
- Good performance if the hardware supports concurrent read/write to the mirrored pair
- Reliable, but expensive



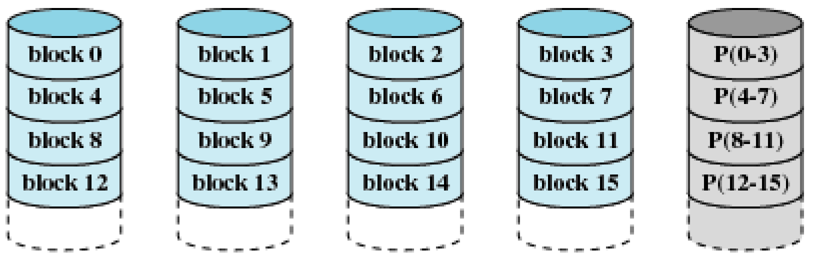
(b) RAID 1 (mirrored)

Parity strip

- Computed and updated at write, verified at read
- Every write results in two read and two write of strips
- A corrupted strip can be recovered

To compute the parity strip...
P(0-3) := b0 ⊕ b1 ⊕ b2 ⊕ b3

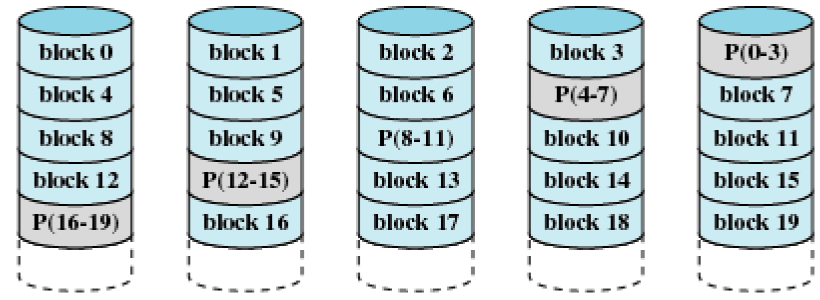
To recover the block 0...
b0 = *P(0-3)* ⊕ b1 ⊕ b2 ⊕ b3



(e) RAID 4 (block-level parity)

RAID 5 (Block-level distributed parity)

- Having all parity strips on one disk may make it a bottleneck. Instead, we can distribute the parity strips among the disks
- If a single disk fails, the system can regenerate the data lost
- Reliable. Good performance with special hardware

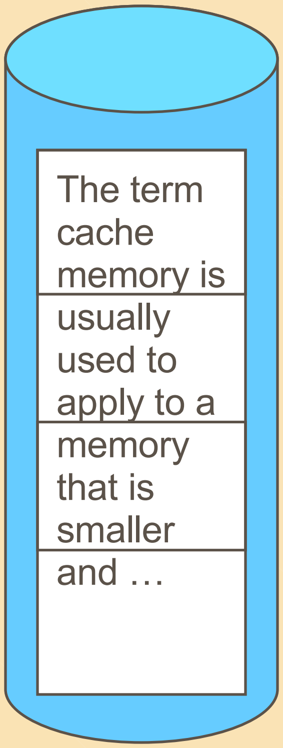


(f) RAID 5 (block-level distributed parity)

Block-oriented disk

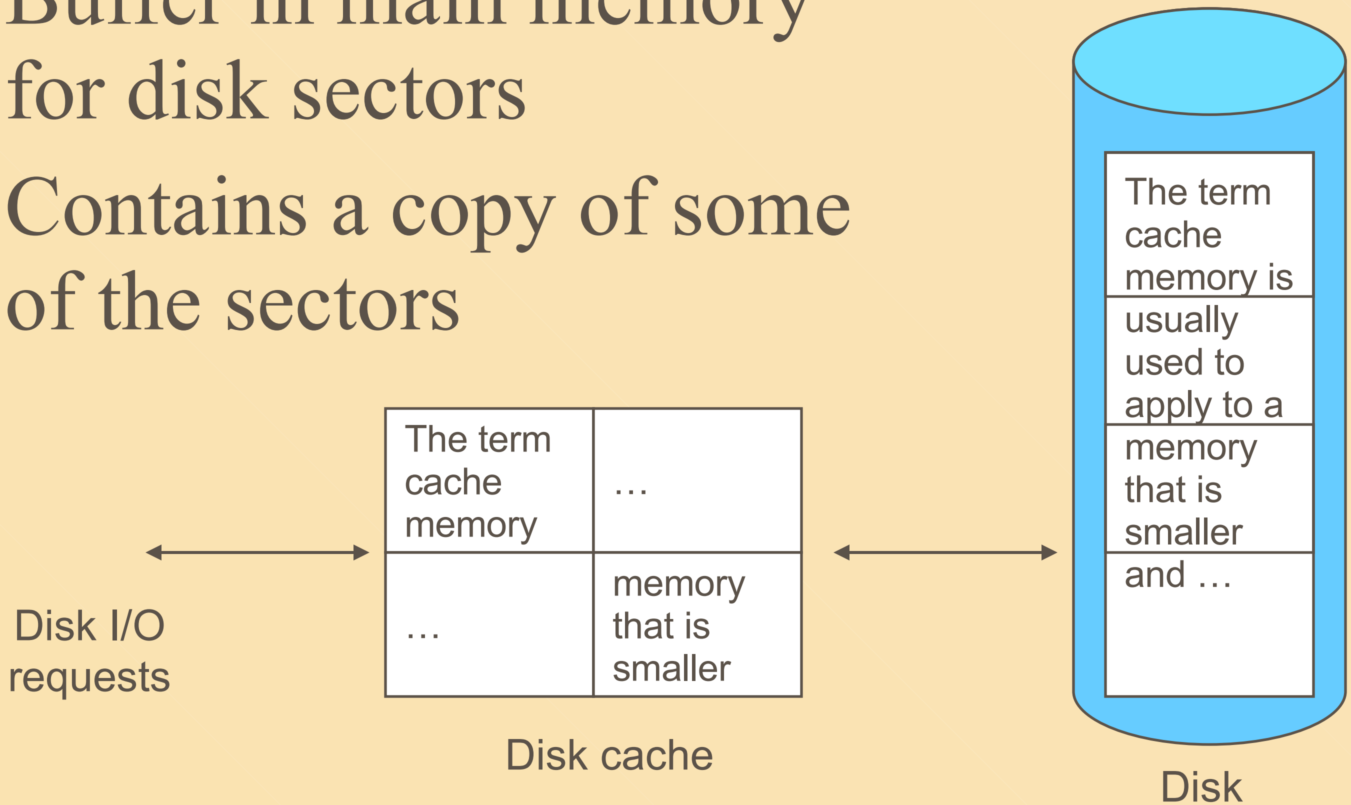
- Disk is block-oriented. One sector is read/written at a time.
- In PC, a sector is 512 byte

```
while (!feof(F)) {
  // read one char
  fscanf(F, "%c", &c);
  …
}
```



Disk Cache

- Buffer in main memory for disk sectors
- Contains a copy of some of the sectors



Disk Cache, Hit and Miss

- When an I/O request is made for a particular sector, the OS checks whether the sector is in the disk cache.
  - If so, (cache hit), the request is satisfied via the cache.
  - If not (cache miss), the requested sector is read into the disk cache from the disk.

Disk Cache, Replacement

- Least Recently Used (LRU)
  - Replace the block that has been in the cache the longest with no reference
- Least Frequently Used (LFU)
  - Replace the block that has experienced the fewest references