

Disk Structure

• logical block

- v 디스크의 외부에서 보는 디스크의 단위 정보 저장 공간들
- v 주소를 가진 1차원 배열처럼 취급
- v 정보를 전송하는 최소 단위

• Sector

- v Logical block이 물리적인 디스크에 대응된 위치
- v Sector 0은 최외곽 실린더의 첫 트랙에 있는 첫 번째 섹터이다.
부팅과 관련된 정보 저장

Disk Management

• Physical formatting (Low-level formatting)

섹터 단위로
나누는 과정

- v 디스크를 컨트롤러가 읽고 쓸 수 있도록 섹터들로 나누는 과정

- v 각 섹터는 header + 실제 data (보통 512bytes) + trailer로 구성
- v header와 trailer는 Sector number, ECC (Error-Correcting Code) 등의 정보가 저장되며
컨트롤러가 직접 접근 및 운영

설계 자체로 훌륭시켜 놓은 것.
Sector의 데이터를 잃을 때 ECC와
실제 data를 확인 시킨 것이 좋다면
bad sector가 아닌 것으로 판단

ECC 규모에 따라서
수정까지 할 수 있고
어려울 경우 예외로 불가능.

상당히 적게 만든 희박한 예 때문에
모든 예외를 잡을 수 있는 것은 아님.
하지만 좋은 해석률을 사용시 상당 부분
이러한 걸 볼 수 있음

• Partitioning

- v 디스크를 하나 이상의 실린더 그룹으로 나누는 과정

- v OS는 이것을 독립적 disk로 취급 (logical disk)

↳ 운영체제는 logical disk!
관심!!

• Logical formatting

- v 파일 시스템을 만드는 것, Swap area 등도 사용 가능.

- v FAT, inode, free space 등의 구조 포함

• Booting

→ CPU의 instruction으로 설명

- v ROM에 있는 "small bootstrap loader"의 실행
- v Sector 0 (boot block)을 load 하여 실행
- v Sector 0은 "full bootstrap loader program"
- v OS를 디스크에 load하여 실행

Disk Scheduling

- Access time의 구성

- Seek time

- 헤드를 해당 실린더로 움직이는데 걸리는 시간 \rightarrow 가장 큰 시班子 구성 요소 (거의 대다수)

한 번의 seek로 많은 양을 읽고 쓰면
정향히 효율적

- Rotational latency \rightarrow 회전하고 있는 플레이트에 헤드가 데이터에 도달하는 시간

- 헤드가 움직이는 속도에 도달하기까지 걸리는 회전 지연 시간

회전하고 있는 플레이트에 헤드가 데이터에 도달하는 시간

- Transfer time

- 실제 데이터의 전송시간
 \rightarrow 광장히 짧은 시간

- Disk bandwidth

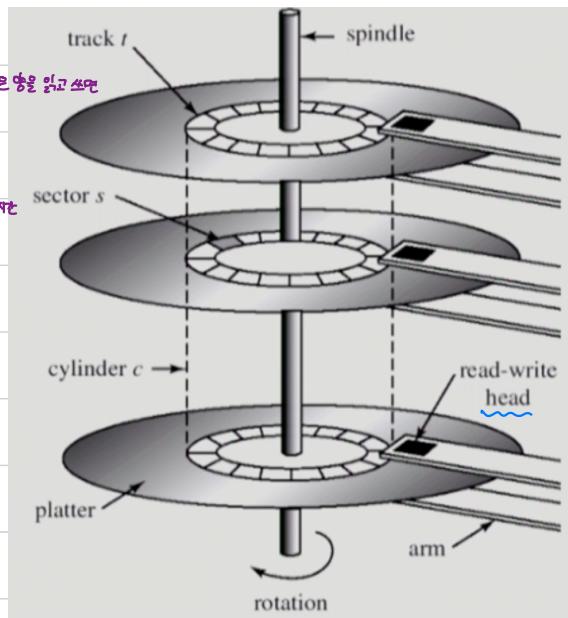
- 단위 시간당 전송된 바이트의 수

- 높아지면 seek 타임 줄여야 함

- Disk Scheduling

- Seek time을 최소화 하는 것이 목표

- Seek time \approx Seek distance



Disk Scheduling Algorithm

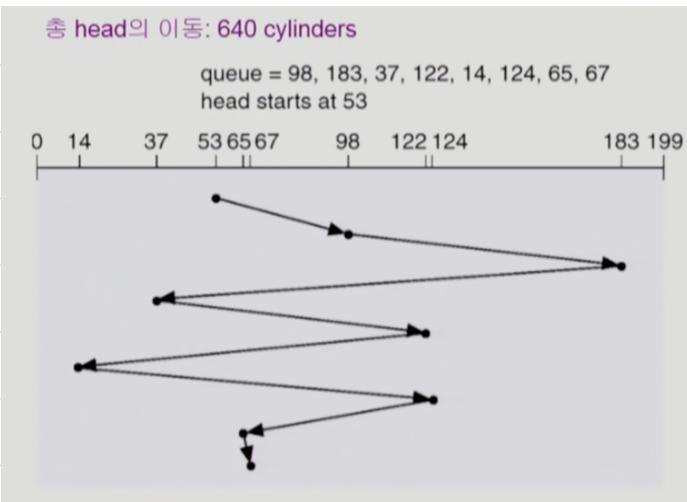
- 큐에 다음과 같은 실린더 위치의 요청이 존재하는 경우 디스크 헤드 53번에서 시작한 각 알고리즘의 수행 결과는? (실린더의 위치는 0~199)

\hookrightarrow 실제로는 실린더의 위치를 정확하게 파악하지 못함
long block을 통해 대략적으로만 안다.

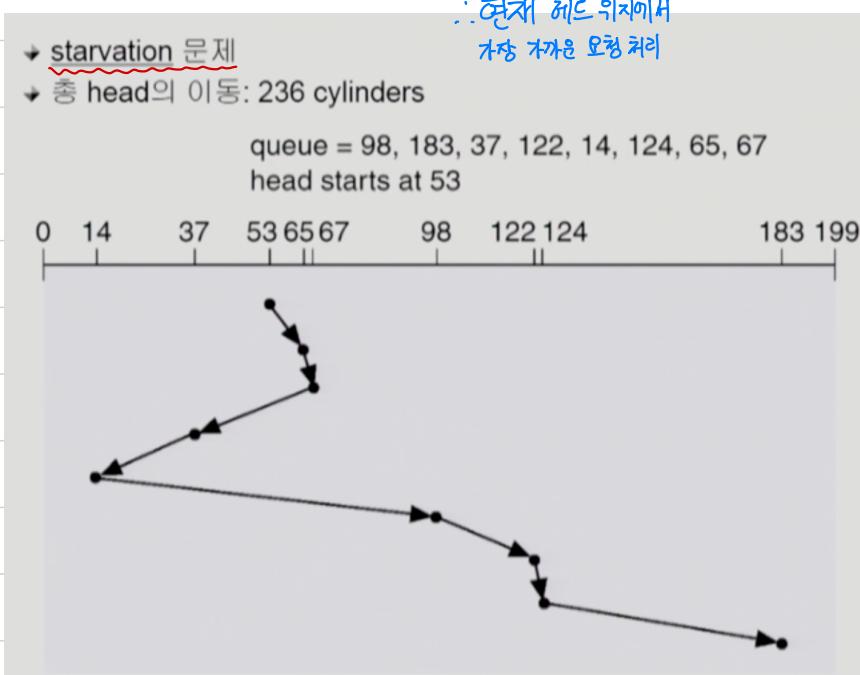
98, 183, 37, 122, 14, 124, 65, 67

- FCFS
- SSTF
- SCAN
- C-SCAN
- N-SCAN
- Look
- C-Look

FCFS (First Come First Served)



SSTF (Shortest Seek Time First)

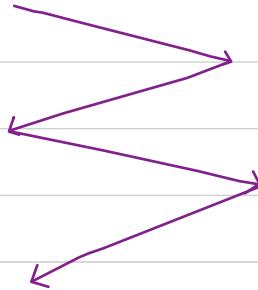


SCAN

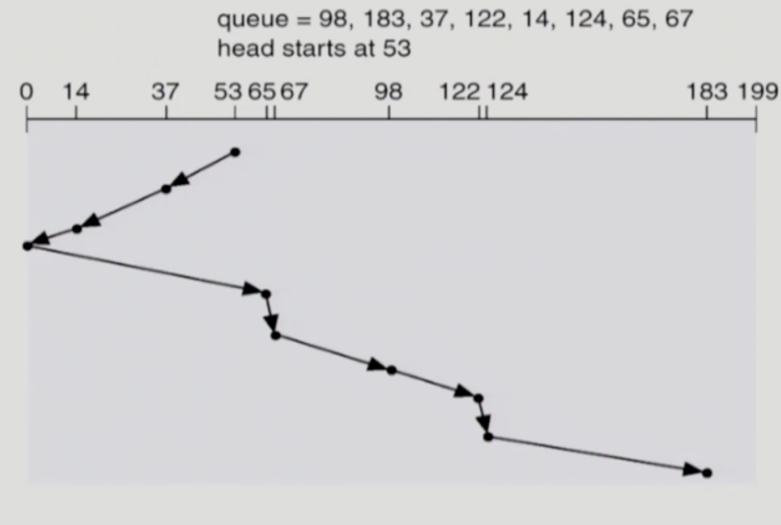
(= 엘리베이터 스케줄링)

→ disk scheduling에 있어 가장 간단하고 확장적인 방법

- DISK 어셈이 디스크의 한쪽 끝에서 다른쪽 끝으로 이동하여 가는 경로에 있는 모든 요청을 처리한다.
- 다른한쪽 끝에 도달하면 역방향으로 이동하여 오는 경로에 있는 모든 요청을 처리하며 다시 반대쪽 끝으로 이동한다.
- 문제점 : 실전터 위치에 따라 대기시간이 다르다.

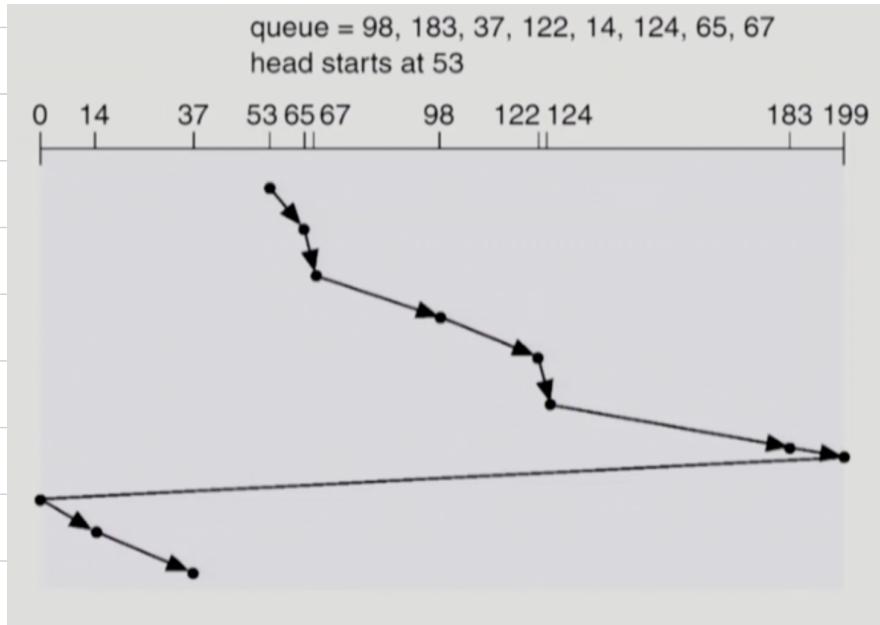
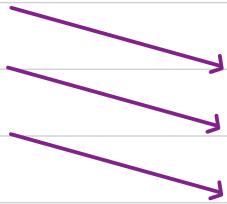


↳ 총 head의 이동: 208 cylinders



C-SCAN

- 헤드가 한쪽 끝에서 다른쪽 끝으로 이동하여 가는 경로에 있는 모든 요청을 처리
- 다른쪽 끝에 도달 했으면 요청을 처리하지 않고 곧바로 출발점으로 다시 이동
- SCAN 보다 균일한 대기 시간을 제공한다.



Other Algorithms

• N-SCAN

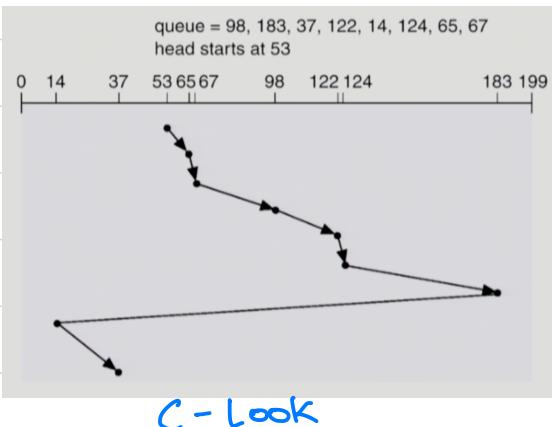
v SCAN의 변형 알고리즘

v 일단 어깨에 한 방향으로 움직이기 시작하면 그 시점 이후에 도착한 job은 되돌아 올때 service

• Look and C-Look

v SCAN이나 C-SCAN은 헤드가 디스크 끝에서 끝으로 이동

v Look과 C-Look는 헤드가 진행 끝이거나 그 방향에서 더 이상 기다리는 요청이 없으면 헤드의 이동 방향을 즉시 반대로 이동한다.



Disk Scheduling Algorithm의 결정

- SCAN, C-SCAN 및 그 응용 알고리즘은 Look, C-Look 등에 일반적으로

디스크 입출력이 많은 시스템에서 효율적인 것으로 알려져 있음

↳ 여러 요청을 한꺼번에 끌어서

- File의 할당 방법에 따라 디스크 요청이 명령을 받음

- 디스크 스케줄링 알고리즘은 필요할 경우 다른 알고리즘으로 쉽게 교체할 수 있도록 OS와 별도의 노드로 작동하는 것이 바람직하다.

Swap-Space Management

• Disk를 사용하는 두 가지 이유

v memory의 volatile한 특성 → file system

v 프로그램 실행을 위한 memory 공간 부족 → swap space (swap area)

• Swap-space

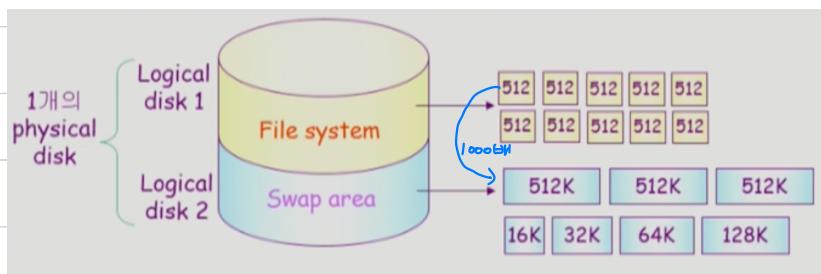
v Virtual memory system에서는 디스크를 memory의 연장 공간으로 사용

v 파일 시스템 내부에 둘 수도 있으나 별도 partition 사용이 일반적

- 공간 효율성이 높다는 속도 효율성이 우선 → 프로세스가 끝나면 잊어버리 내용 아니니까

- 일반 파일보다 훨씬 많은 시간은 시간은 문제하고 자주 참조됨

- 파일, block의 크기 및 저장 방식이 일반 파일 시스템과 다름



RAID

- RAID(Redundant Array of Independent Disk)

- v 여러개의 디스크를 묶어서 사용
제한된 디스크

- RAID의 사용 목적

- v 디스크 처리 속도 향상
 - 여러 디스크에 block의 내용을 분산저장
 - 병렬적으로 읽어들 (Interleaving, Striping)
- v 신뢰성 (Reliability) 향상
 - 동일 정보를 여러 디스크에 중복 저장
 - 하나의 디스크가 고장(failure) 시 다른 디스크에서 읽어옴 (Mirroring, shadowing)
 - 단순한 중복 저장이 아니라 일부 디스크가 parity를 저장하여 공간의 효율성을 높일 수 있다.

