Disk Management & Scheduling

- Disk Strutcture (디스크 구조)
    - Sector
        - 디스크 관리하는 최소의 단위
        - Sector 0은 최외곽 실린더의 첫 트랙에 있는 첫 번째 섹터
            - 부팅과 관련된 정보가 저장
    - logical block
        - 디스크 외부에서 보는 디스크의 단위 정보 저장 공간들
            - 이 block이 sector에 mapping해서 들어가 있는 것.
        - 1차원 배열처럼 취급
            - 배열 몇 번째에 있는 원소 달라는 요청
        - 디스크 컨트롤러는 어떤 블록에 어떤 조각에 위치 시킬 것인지를 결정
        - 정보를 전송하는 최소 단위
- Disk Management
    - Physical Formatting(Lowlevel Formatting)
        - 디스크를 컨트롤러가 읽고 쓸 수 있도록 섹터들로 나누는 과정
        - logical block외 부가적으로 저장할 수 있는 공간 따라 붙음.
            - Header와 Trailer
                - 내용이 잘 저장되었는지 확인하는 error correction code(ECC)
                    - error의 정도에 따라서 data의 수정도 가능
                - 주소 mapping을 위한 sector 번호
                - 데이터 읽고 쓰기 위한 부가적인 정보!
    - Partitioning
        - 디스크를 하나 이상의 실린더 그룹으로 나누는 과정 (C, D 드라이브)
        - OS는 이것을 독립적 disk로 취급(logical disk)
    - Logical Formatting
        - 파일 시스템을 만드는 것.
        - FAT, inode, free space 등의 구조 포함
    - Booting
        - ROM
            - 메모리 영역 중 전원이 나가더라도 내용이 유지되는 구조
            - Booting을 위한 loader가 저장됨.(small bootstrap loader)
                - 하드디스크에서 sector0(boot block) 메모리에 올리고 실행하라고 지시
                - 운영체제 커널을 메모리에 올려서 실행
- Disk Scheduling
    - Access time의 구성
        - Seek time
            - 헤드를 해당 실린더로 움직이는 데 걸리는 시간

        - 가장 큰 시간 구성 요소
        - Disk Scheduling은 seek time 최소화하는 것이 목표
      - Rotational latency
        - 헤드가 원하는 섹터에 도달하기 까지 걸리는 회전지연시간
      - Transfer time
        - 실제 데이터 전송 시간
    - Disk bandwidth
      - 단위 시간 당 전송된 바이크의 수
- Disk Scheduling Algorithm
    - 위에 다음과 같은 실린더 위치의 요청이 존재하는 경우 디스크 헤드 53번에서 시작한 각 알고리즘의 수행 결과는? (실린더 위치는 0-199)
      - 98, 183, 37, 122, 14, 124, 65, 67
    - FCFS(First Come First Service)
      - 온 순서대로
    - SSTF(Shortest Seek Time First)
      - 제일 가까운 요청을 가장 먼저 처리
      - Starvation 문제
        - 가까운 주소가 또 들어오면 계속 처리를 못함
        - 높은 주소 요청이 영원히 처리 안 될 수 있음
      - (현재 53 가정)
        - 53 -> 65 -> 67 -> 37 -> 14 -> 98 -> 122 -> 124 -> 183
    - SCAN
      - disk aim이 디스크 한쪽 끝에서 다른쪽 끝으로 이동하며 가는 길에 있는 모든 요청 처리
      - 실린더 위치에 따라 대기 시간이 다르다.
        - 가장자리 이미 Head가 떠나면 오래 기다려야 함.
        - 53 -> 37 -> 14 -> 0 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199
    - C-SCAN
      - 헤드가 한쪽 끝에서 다른 끝으로 이동하며 가는 길에 있는 모든 요청 처리
      - 다른쪽 끝에 도달했으면 요청 처리하지 않고 출발점으로 이동
      - SCAN보다 균일한 대기시간 제공
        - 이동거리는 조금 길어질 수 있음
      - 53 -> 65 -> 67 -> 98 -> 122 -> 124 -> 183 -> 199 -> 0 -> 14 -> 37
    - 기타
      - N-SCAN
        - 지나가는 도중에 들어온 것은 다음 번에 처리
      - LOOK, C-LOOK
        - SCAN, C-SCAN 요청이 있던 없던 이동
        - 그 방향에 더 이상 요청이 없으면 방향을 바꿈.
    - 결정
      - SCAN에 기반한 알고리즘 많이 씀 (디스크 입출력이 많은 경우 효율적)
      - File의 할당 방법에 따라 디스크 요청이 영향을 받음
        - 연속적 vs 산발적...

        - 필요할 경우 다른 알고리즘으로 설계 교체할 수 있도록 OS와 별도의 모듈로 작성되는 것이 바람직
- Swap Space Management
    - Disk를 사용하는 두 가지 이유
        - memory의 volatile한 특성 (휘발성) -> file system
        - 프로그램 실행 위한 공간 부족 -> swap space (swap area)
    - Swap-Space
        - Virtual memory system에서는 디스크를 memory의 연장공간으로 사용
        - 파일 시스템 내부에 둘 수도 있으나 별도의 partition 사용이 일반적
            - 공간 효율성보다는 속도 효율성
                - 프로세스 끝나면 없어질 내용
            - 일반 파일보다 훨씬 짧은 시간만 존재하고 자주 참조됨.
            - block의 크기 및 저장 방식이 일반 파일시스템과 다름
        - 큰 단위를 순차적으로 할당
- RAID (Redundant Array of Independent Disk)
    - 여러개의 디스크 묶어서 사용
    - 목적
        - 디스크 처리 속도 향상
            - 여러 디스크에 block의 내용을 분산 저장
            - 병렬적으로 읽어 옴 -> 빨라짐 (Interleaving, striping)
        - 신뢰성 향상
            - 동일 정보를 여러 디스크에 중복 저장
            - 하나의 디스크가 고장시 다른 디스크에서 읽어옴 (Mirroring, shadowing)
            - 단순한 중복저장이 아니라 일부 디스크에 parity를 저장하여 공간의 효율성을 높일 수 있다.
                - 오류 생겼는 지 알아내고 복구 할 수 있을 정도로만 저장. (Hash)