File System

File

- 일반적으로 비휘발성의 보조장치에 저장
- 운영체제는 다양한 저장 장치를 file이라는 동일한 논리적 단위로 볼 수 있게 해줌
- Operation - create, read, write, reposition(lseek), delete, close등 이 가능

File Attribute (File의 metadata)

- 파일 자체의 내용이 아니라 파일을 관리하기 위한 각종 정보들
    - 파일 이름, 유형, 저장된 위치, 사이즈
    - 접근권한, 시간, 소유자 등

File System

- 운영체제에서 파일을 관리하는 부분
- 파일 및 파일의 메타데이터, 디렉토리 정보 등을 관리
- 파일의 저장 방법 결정
- 파일 보호 등

Directory

- 파일의 메타데이터 중 일부를 보관하고 있는 일종의 특별한 파일
- 그 디렉토리에 속한 파일 이름 및 파일 attribute들
- operation
    - 검색, 생성, 삭제, 정렬, 이름 변경 등

Partition(Logical Disk)

- 하나의 물리적 디스크 안에 여러 파티션을 두는게 일반적
- 여러개의 물리적인 디스크를 하나의 파티션으로 구성하기도 함
- 디스크를 파티션으로 구성한 뒤 각각의 파티션에 file system을 깔거나 swapping 등 다른 용도로 사용할 수 있음

Open

파일의 메타데이터를 메모리로 올려놓는 것

파일시스템에 메타데이터, 내용이 저장되어 있는데 오픈하게 되면 메타데이터가 메모리로 올라옴

- 사용자 메모리 영역에서 시스템 콜 (오픈해라)
- 커널메모리 영역에서 root의 metadata를 open file table에 올려놓음(루트를 오픈함)
- 루트의 컨탠츠에서 a의 메타데이터를 찾아 그거를 open file table에 올려놓음
- a의 컨탠츠에서 b의 메타데이터를 찾아 그거를 open file table에 올려놓음
- 각 프로세스마다 오픈한 파일들에 대한 메타데이터 포인터를 가지고 있는 배열이 만들어지는데 b의 메타데이터가 가르키는 포인터의 인덱스를 사용자에게 리턴된다.
- 그걸 read하면 b의 메타데이터에서 찾은 b의 content를 읽어서 운영체제가 메모리 공간 일부에 먼저 카피해서 갖다놓은 다음 사용자에게 전달

File Protection

각 파일에 대해 누구에게 어떤 유형의 접근을 허락할 것인가?

Access control 방법

1. Access control Matrix
    - 행렬에 각각의 사용자에게 어떤 권한이 있는지 표시
    - Access control list: 파일별로 누구에게 어떤 접근 권한이 있는지 표시
    - Capability: 사용자 별로 자신이 접근 권한을 가진 파일 및 해당 권한 표시
2. Grouping
    - 전체 유저를 owner, group, public의 세 그룹으로 구분
    - 각 파일에 대해 세 그룹의 접근권한을 3비트씩으로 표시
    - ex) UNIX
3. Password
    - 파일마다 패스워드를 두는 방법
    - 모든 접근 권한에 대해 하나의 password: all-or-nothing
    - 접근 권한별 password: 암기 문제, 관리문제

File system의 Mounting

하나의 물리적 디스크를 파티션을 통해 논리적 디스크로 나눌 수 있고 각각의 파티션에 파일시스템을 설치 가능

만약 다른 파티션에 설치된 파일 시스템에 접근해야 한다?

Mounting 연산 활용

루트 파일시스템의 특정 디렉토리 이름에 또다른 파티션에 있는 파일 시스템을 마운트롤 해주면 마운트된 디렉토리를 접근하게 되면 또 다른 파일 시스템의 루트 디렉토리에 접근하는 것과 같다.

Access Methods

시스템이 제공하는 파일 정보의 접근 방식

1. 순차 접근
    - 카세트 테이프를 사용하는 방식처럼 접근
    - 읽거나 쓰면 offset은 자동적으로 증가
2. 직접 접근

- LP레코드 판과 같이 접근하도록 함
- 파일을 구성하는 레코드를 임의의 순서로 접근할 수 있음

Allocation of File Data in Disk

Continguous Allocation

연속할당

ex) 크기가 6이다 2-8까지 연속해서 블록에 할당해주는것

장점: Fast I/O

- 한번의 seek/rotation으로 많은 바이트 transfer
- Realtime file용으로 또는 이미 run 중이던 process의 swapping 용

Direct Access(=random access) 가능

단점: 중간 중간의 프리블럭(내용이 들어있지 않은 블럭)의 크기가 균일하지 않다 /

파일의 크기를 키우는데 제약이 있다.

- file 생성시 얼마나 큰 hole을 배당할 것인가?
- grow 가능 vs 낭비(internal fragmentation)

Linked Allocation

장점: External Fragmentaion 발생 x

단점:

- No random access
- Reliability문제
  - 한 sector가 고장나 pointer가 유실되면 많은 부분을 잃음
- Pointer를 위한 공간이 block의 일부가 되어 공간 효율성을 떨어트림
  - 512 bytes/sector, 4bytes/pointer

변형

File allocation table(FAT) 파일 시스템

- 포인터를 별도의 위치에 보관하여 reliability와 공간 효율성 문제 해결

Indexed Aloocation

directory에 파일의 위치정보를 바로 저장하는게 아니라 인덱스의 위치(인덱스 블럭)을 가르키게 되어있음.

ex) jeep의 인덱스 블럭이 19번이고 19번 블럭에 jeep이 저장되어있는 위치 정보가 있음

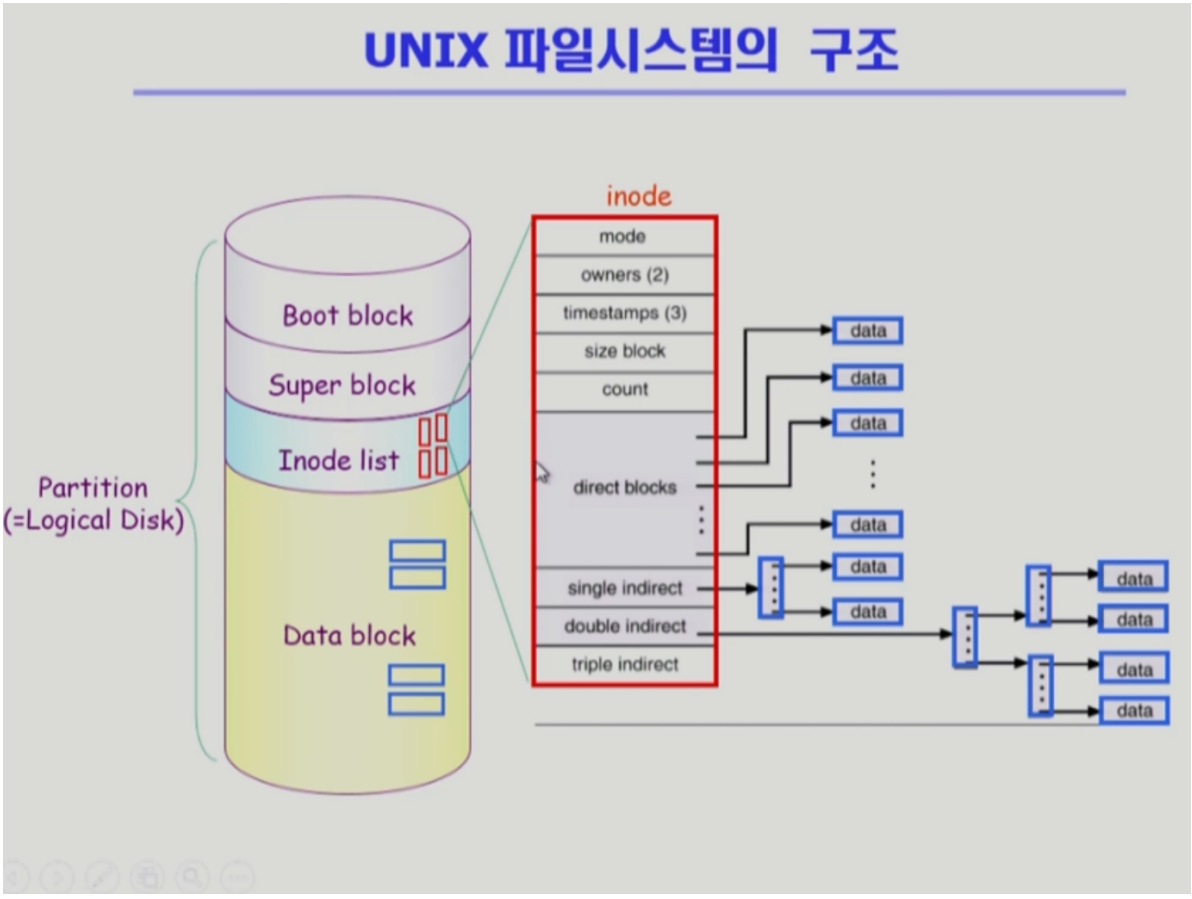
장점:

- External Fragmentation이 발생하지 않음
- Direct Access 가능

단점:

- 작은 파일의 경우 공간 낭비
- 너무 큰 파일의 경우 하나의 block으로 인덱스를 저장하기에 부족
  - 해결 방안
  - linked scheme
  - multi-level-index

Unix 파일시스템의 구조



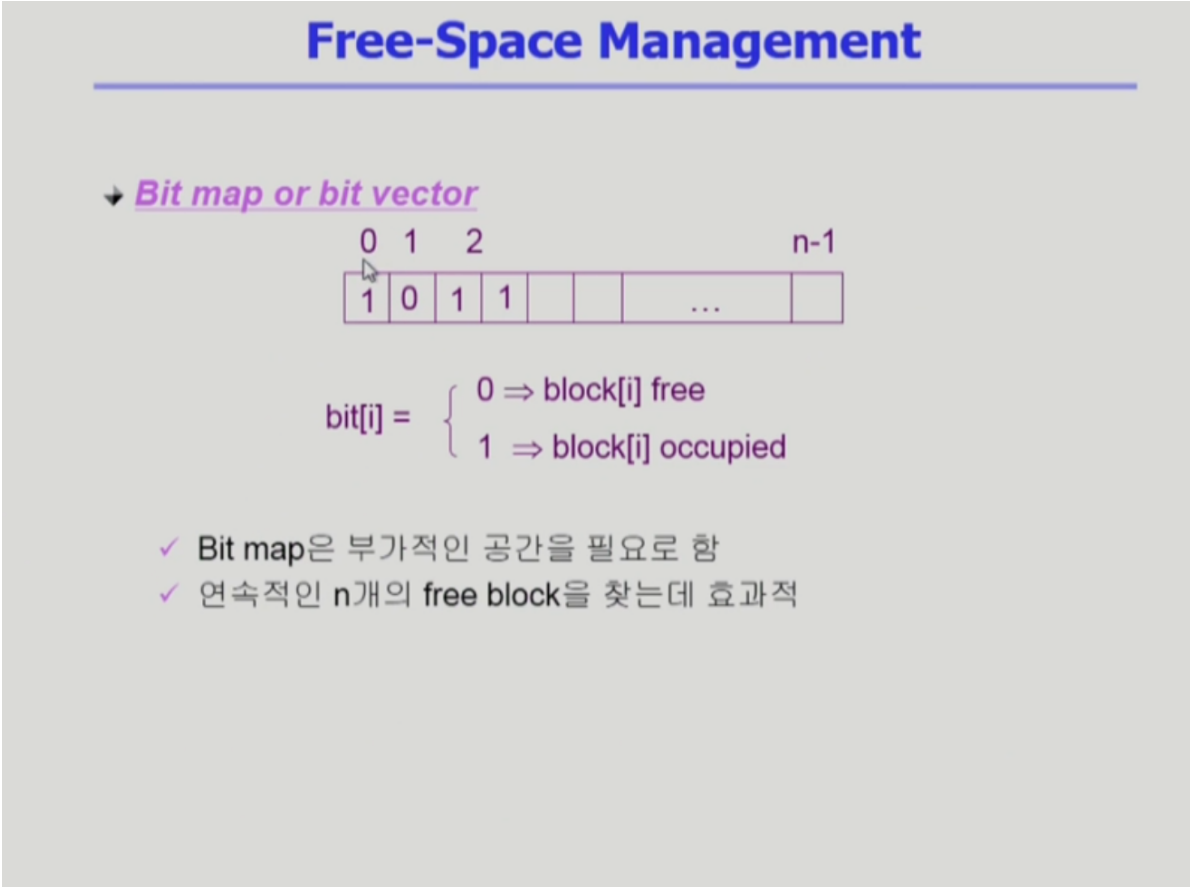
Linked Allocation 적용

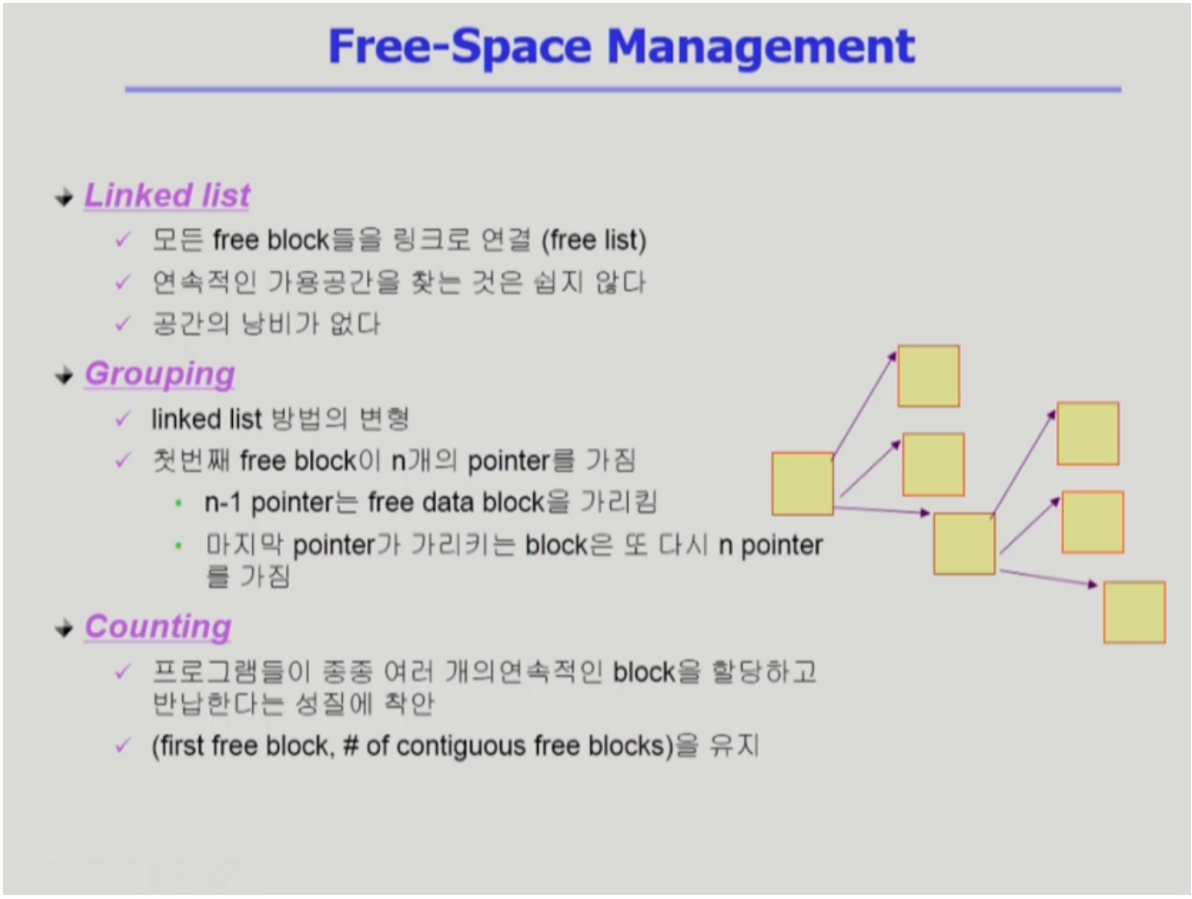
다음위치를 위해 오직 FAT만 확인해보면 된다.

FAT은 보통 디스크에 2카피 이상을 저장하고 있다. ==> Reliable요소 개선

Free space Management

Bit map or bit vector





Directory Implementation

Linear List

File 이름과 메타데이터의 list

구현이 간단하다

디렉토리 내에 파일이 있는지 찾기 위해서는 linear search가 필요

Hash Table

linear List + hashing

해쉬 테이블은 파일 이름을 이 파일의 리니어 리스트 위치로 바꾼다

서치타임을 없앰

collision 발생 가능

File Metadata의 보관 위치

- 디렉토리 내에 직접 보관
- 디렉토리에는 포인터를 두고 다른 곳에 보관
    - inode, FAT 등

Long File Name의 지원

- 파일 이름, 파일의 메타데이터의 리스트에서 각 entry는 일반적으로 고정 크기
- file name이 고정 크기의 entry 길이보다 길어지는 경우 entry의 마지막 부분에 이름의 뒷부분이 위치한 곳의 포인터를 두는 방법
- 이름의 나머지 부분은 동일한 디렉토리 파일의 일부에 존재

VFS AND NFS

VFS(virtual file system)

- 서로 다른 다양한 파일시스템에 대해 동일한 시스템 콜 인터페이스(API)를 통해 접근할 수 있게 해주는 OS의 layer

NFS(network file system)

- 분산 시스템에서는 네트워크를 통해 파일이 공유될 수 있음
- NFS는 분산환경에서의 대표적인 공유 방법