파일 시스템

파일 시스템

1. 파일 시스템

파일이란

- 논리적인 저장 단위로, 관련된 정보 자료들의 집합에 이름을 붙인 것
- 일반적으로 레코드 혹은 블록 단위로 비휘발성 보조기억장치에 저장

구조

- 메타데이터 : 데이터 영역에 기록된 파일의 이름, 위치, 크기 ,시간정보, 삭제유무 등 파일의 정보
- 데이터 영역 : 파일의 데이터

File Attribute (MetaData)

- 파일을 관리하기 위한 각종 정보
- Name : 사람이 읽을 수 있는 형태의 유일한 정보
- Identifier : 파일 시스템 내부의 유일한 식별 태그(이름)
- Type : 다양한 종류의 File을 지원하기 위해 필요
- Location : 장치내에서 파일의 위치를 위한 포인터
- Size : 현재 파일의 크기
- Protection : 읽기, 쓰기, 실행에 대한 권한 제어
- Time, date, and user identification : 파일 사용에 대한 보호, 보안, 모니터링을 위한 데이터

파일 시스템

정의

- 컴퓨터에서 파일이나 자료를 쉽게 발견할 수 있도록 유지 / 관리하는 방법
- 저장매체에는 수많은 파일이 있기 때문에 이런 파일들을 고나리하는 방법

특징

- 커널 영역에서 동작
- 파일 CRUD 기능을 원할히 수행하기 위한 목적
- 계층적 Directory 구조를 가짐
- 디스크 파티션 별로 하나씩 둘 수 있음

역할

- 파일 관리 (CRUD)
- 보조 저장소 관리
- 파일 무결성 매커니즘
- 접근 방법 제공

개발 목적

- 하드 디스크와 메인 메모리 사이의 속도 차이를 줄이기 위함
- 파일 관리
- 하드 디스크의 용량을 효율적으로 이용하기 위함

파티션

- 연속된 저장 공간을 하나 이상의 연속되고 독립적인 영역으로 나누어 사용할 수 있도록 정의한 규약
- 하나의 물리적 디스크 안에 여러 파티션을 두는 게 일반적이다.
- 하지만 여러 물리적 디스크를 하나의 파티션으로 구성하기도 한다.

2. 접근 방법

1. 순차 접근 파일 (Sequential Access File)

> 입력되는 데이터가 물리적으로 연속적인 위치에 기록된느 방식

- 파일의 내용이 하나의 연속된 줄 형태로 기록
- 연속적 데이터 접근을 기반한 매체(자기 테이프)에서 가장 효율적임
- 장점
  - 저장 효율이 높음
  - 낭비되는 저장 공간이 없음
  - 데이터의 빠른 읽기/쓰기가 가능
- 단점
  - 내용 갱신 및 주간 삽입이 비효율적인
  - 위치 이동 시 너무 오래 걸림

2. 직접 접근 파일 (Direct Access File)

> 물리적인 저장매체의 주소에 직접 접근하는 방식

- 전문 지식 필요
- 해시 기법 사용
- Mapping 함수와 표가 필요
- 장점
  - 블록의 개수가 많더라도 빠른 접근이 가능
  - 어떤 레코드라도 평균 접근 시간 내에 접근 가능
- 단점
  - 해시 충돌에 대한 해결방안이 필요
  - 키 변환법에 따라 공간의 낭비를 가져올 수 있음

3. 인덱스 접근 파일 (Indexed Sequential Access File)

> 순차 파일 구조에 인덱스를 위한 필드를 추가한 방식

- 주로 디스크 기반의 저장매체에서 사용
- 장점
  - 빠른 접근 및 위치 이동이 가능
  - 중간에 삽입 및 변경이 쉬움
- 단점
  - 내부 단편화가 발생
  - 디스크의 낭비가 발생될 수 있음
  - 인덱스를 처리하는 추가적인 시간이 소모
  - 인덱스 저장 공간 및 오버플로 처리를 위한 별도 공간 필요

3. 디스크 할당

디스크 공간 할당과 회수는파일의 데이터를 물리적인 디스크에 저장하고 삭제하는 방법을 의미

데이터의 물리적인 연속성에 따라 두 가지로 구분

블록

- 하드디스크와 컴퓨터 사이 데이터 전송을 위한 단위
- 한 블록은 하드디스크내 여러 개의 섹터로 구성됨
- 연속 블록 할당(Contiguous Block Allocation)
- 불연속 블록 할당(링크 블록 할당)
- – 섹터 단위형
- – 블록 단위형

연속 블록 할당(Contiguous Block Allocation)

- 물리적으로 연속적인 공간에 저장

- 저장할 크기(공간)을 미리 지정
- 파일마다 크기가 다르기 때문에, 잦은 추가/삭제 발생시 단편화 발생
- 주기적인 통합(Coalescing), 집약(Compaction, Garbage Collection) 작업 필요

파일명	시작 블록	블록 개수
A.TXT	3	6
B.TXT	10	5
C.TXT	19	10

0	1	2	3	4	5	6	7	8	9
10	11	12	13	14	15	16	17	18	19
20	21	22	23	24	25	26	27	28	29
30	31	32	33	34	35	36	37	38	39
40	41	42	43	44	45	46	47	48	49
50	51	52	53	54	55	56	57	58	59

불연속 블록 할당(링크 블록 할당)

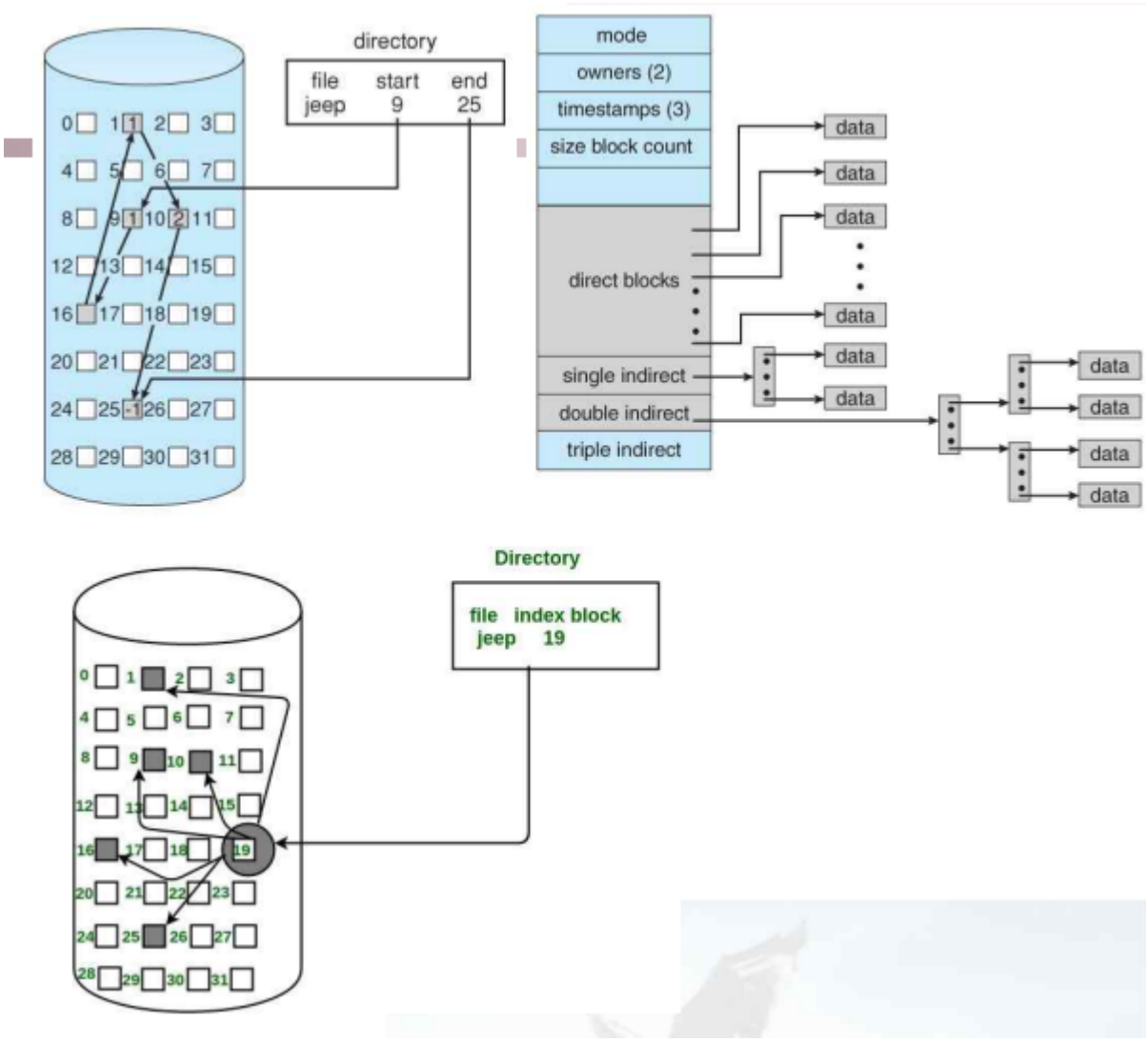
- 디스크 공간을 일정한 길이를 갖는 단위(섹터, 블록)으로 나누어 할당
- 분할된 영역은 독립적으로 취급
- 파일의 데이터들은 분할된 영역에 저장
- 저장하는 공간 외에 분할된 정보와 파일의 연결된 데이터 정보를 저장해야 하는 영역이 부가적으로 필요

  섹터 단위형

  - 파일의 데이터가 섹터 단위로 분산되어 저장
  - 섹터는 연결 리스트(Linked-List) 구조 형태로 연결
  - 파일의 크기가 작아지면 사용하지 않는 섹터는 반환됨

블록 단위형

- 블록 체인 기법 – 여러 개의 섹터를 묶은 블록을 체인처럼 연결
- 인덱스 블록 체인 기법(index block chaining)
  - – 인덱스에 블록의 주소를 링크
  - – 예: UNIX 파일 시스템
- 블록 단위 파일 사상 기법
  - – 파일 정보의 해당 블록을 사상시켜 연결
  - – 예: MS-DOS, MS-Windows



운영체제 별 파일 시스템

Windows

FAT(File Allocation Table)

- DOS때부터 사용하고 있는 대표적인 파일 시스템
- 하드디스크에 FAT 영역을 만들어 파일의 실제 위치 등의 정보를 기록해 두고 이를 이용
- 장점

    - 호환성 우수
    - 단순성
    - 저용량 볼륨에 최적화
- 단점
    - 보안에 취약 (실제 하드에 파일 정보 저장)
    - 대용량 볼륨의 비효율적 이용 (공간을 추가로 사용해야 됨)

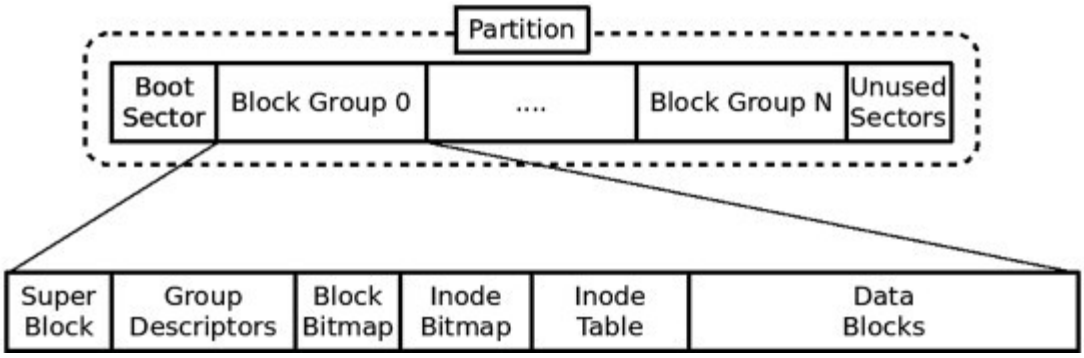
NTFS

- MFT(Master File Table)를 사용하여 관리하며 Mirror와 파일로그가 유지되어 비상 시 파일 복구가 가능함
- FAT 기능에 대용량 하드 디스크를 지원, 보안기능, 압축 기능, 원격 저장소 기능 등을 추가하여 만든 윈도우 NT 전용 파일 시스템
- 장점
    - 대용량 볼륨 지원
    - 디스크 효율적 사용
    - 강력한 보안 기능
    - 자동 압축 및 안정성
    - 향상된 파일 이름 저장 및 길이 지원
- 단점
    - 다른 운영체제에 호환 불가
    - 저용량 볼륨에서 FAT 대비 속도가 느림

Linux

EXT(Extended File System)

- 단일 트리구조를 갖는다
- 하나의 root 디렉토리가 있고, 디렉토리 - 하위 디렉토리 - 데이터 파일로 구성



Mac OS

APFS

Unix

UFS(Unix File System)

- Tree 구조로 이루어져 있음
- i-node에 파일에 대한 모든 정보를 저장

디렉토리 구조

정의

- Directory는 파일의 메타데이터 중 일부를 보관하고 있는 일종의 특별한 파일
- 해당 디렉토리에 속한 파일 이름과 속성들을 포함하고 있다.

기능

- Search
- Create
- Delete
- List
- Rename

- Traverse

구성

- Efficiency : 디렉토리는 파일을 빠르게 탐색할 수 있어야 한다.
- Naming : 적절한 이름으로 사용자들이 편리하게 사용할 수 있으면 좋을 것이다.
- Grouping : 파일들을 적절한 분류로 그룹화 해두면 사용하기 편리할 것이다.

레벨

1단계 디렉토리 : Single Level

- 가장 간단한 구조 모든 파일들이 디렉터리 밑에 존재하는 형태
- 파일들은 서로 유일한 이름을 갖는다.
- 서로 다른 사용자라도 같은 이름의 파일들을 사용할 수 없다.

단점

- 파일이 많아지거나 다수의 사용자가 사용하는 시스템이라면 문제 발생
- Naming Problem
- Grouping Problem : 시스템에 오직 1개의 디렉토리만 있기 때문에 그룹핑 불가능

2단계 디렉토리 : Two Level

- 각 사용자별로 별도의 디렉터리를 갖는 형태
- 서로 다른 사용자들은 동일한 파일의 이름을 사용할 수 있다.
- 효율적인 탐색 가능 : 각 파일들은 경로 명으로 찾아가야 함
- 종류
    - UFD(User) : 자신만의 사용자 파일 디렉터리
    - MFD(Meat) : 사용자의 이름과 계정 번호로 색인되어 있는 디렉토리
      각 엔트리는 사용자의 UFD를 가리킨다.
- 그룹핑 기능은 존재하지 않음

Tree Structured Directories

- 사용자들이 자신의 서브 디렉터리를 만들어서 파일 구성 가능
- 각 사용자는 하나의 루트 디렉터리를 가지면 모든 파일은 고유한 경로를 갖는다. (절대, 상대)
    - 절대 경로 : 파일을 읽기 위한 파일 전체 경로 (root부터)
    - 상대 경로 : 현재 위치를 기준으로 하여 목적지까지의 상대적인 경로
- 해당 경로를 통하여 효율적인 탐색 및 그룹화가 가능
- 디렉터리 또한 일종의 파일이기 때문에 일반 파일인지 디렉토리인지 구분할 필요가 있음 (bit)

Acyclic Graph (비순환 그래프) Directory

- 디렉터리들이 서브 디렉터리들과 파일을 공유할 수 있도록 한다.
- 서로다른 사용자가 동일한 파일에 대해서 서로 다른 Alias(별칭)을 사용 가능하다.
- 파일을 무작정 삭제하게 되면 현재 파일을 가리키는 포인터는 대상이 사라지게 된다
    - 해당 포인터를 Dangling Pointer라고 한다.
- 참조되는 파일에 참조 계수를 두어 참조 계수가 0이 되면 파일을 참조하는 링크가 존재하지 않는다는 의미이므로 그때 파일을 삭제할 수 있도록 한다.

General Graph (일반 그래프) Directory

- 순환을 허용하는 그래프 구조이다.
- 순환이 허용되면 무한 루프에 빠질 수 있기 때문에 잘 쓰이지 않는다.
- 순환이 발생하지 않도록 하위 디렉터리가 아닌 파일에 대한 링크만 허용하거나 Garbage Collection을 통해 전체 파일 시스템을 순회하고 접근 가능한 모든 것을 표시한다.

자원 보호 기법

운영체제에서 자원을 보호하는 기법

자원 보호 기법 (접근 제어 기법)

- 접근 제어 행렬 (ACM)

- 접근 제어 리스트 (ACL)
- 자격 리스트 (CL)

접근 제어 행렬(Access Control Matrix)

- 객체의 사용 권한을 모든 사용자 리스트와 함께 표로 관리
- 사용자가 많아지면 공간 낭비가 발생 ⇒ 효율성 하락

	CPU	메모리	디스크	프로세스	USB	파일1
관리자	RWX	RWX	RWX	RWX	RWX	RWX
사용자1	R-X	R-X	R-X	R-X	R-X	R-X
사용자2	RW-	RW-	RW-	R--	R--	R--
사용자3	R--	RW-	RW-	R--	R--	R--

접근 제어 리스트 (Access Control List)

- 자원 영역별로 권한을 구성
- 리스트내 권한이 있는 사용자만 나열 ⇒ 공간 절약

	접근 가능자
CPU	시스템 관리자(RWX), 사용자1(R-X), 사용자2(---)
메모리	시스템 관리자(RWX), 사용자1(R-X)
디스크	시스템 관리자(RWX), 사용자1(RW-)
프로세스	시스템 관리자(RWX)

자격 리스트 (Capacity List)

- 사용자별 접근 가능한 자원 항목들을 정의
- 접근 제어 행렬에서 각 행만 따온 것
- 각 권한은 객체와 그 객체에 허용된 연산자로 구성

	시스템 관리자
CPU	RWX
메모리	RWX
디스크	RWX
프로세스	RWX

	사용자 1
CPU	R-X
메모리	R-X
디스크	R--
프로세스	-WX

	사용자 2
CPU	R-X
메모리	R--
디스크	RW-
프로세스	--X