파일시스템

파일과 파일시스템

- **파일** : **논리적인 저장 단위**로 **관련된 정보 자료들의 집합에 이름을 붙인 것**

  컴퓨터 시스템의 편리한 사용을 위해 정보 저장의 일관된 논리적 관점을 제공한다. 일반적으로 레코드 혹은 블록 단위로 비휘발성 보조 기억장치에 저장된다.

  - **파일 속성 (File attribute)** 또는 **파일의 메타데이터(metadata)** : 파일을 관리하기 위한 각종 정보

    파일 자체의 내용은 아니며 파일의 이름, 유형, 저장된 위치, 파일 사이즈, 접근 권한, 소유자, 시간 등 파일에 대한 전반적인 정보를 말한다.

- **파티션** : **연속된 저장 공간을 하나 이상의 연속되고 독립적인 영역으로 나누어 사용할 수 있도록 정의한 규약.** 하나의 물리적 디스크 안에 여러 파티션을 두는 게 일반적이지만, 여러 물리적 디스크를 하나의 파티션으로 구성하기도 한다.
- **파일 시스템** : **운영체제와 모든 데이터, 프로그램의 저장과 접근을 위한 기법을 제공**한다. 시스템 내의 모든 파일에 관한 정보를 제공하는 **계층적 디렉터리 구조**이고, 파일 및 파일의 메타데이터, 디렉터리 정보 등을 관리한다.

  - 구조
    - 메타 영역 : 파일의 메타 데이터
    - 데이터 영역 : 파일의 데이터
  - 특징
    - 커널 영역에서 동작한다.
    - 파일 CRUD 기능을 원활히 수행하기 위한 목적이다.
    - 디스크 파티션 별로 하나씩 둘 수 있다.

- 역할
  - 파일 관리
  - 보조 저장소 관리
  - 파일 무결서 메커니즘
  - 접근 방법 제공

접근 방법

1. 순차 접근(Sequential Access)

파일의 정보가 레코드 순서대로 처리되는 가장 간단한 접근 방법이다. 대부분의 연산은 read와 write이다

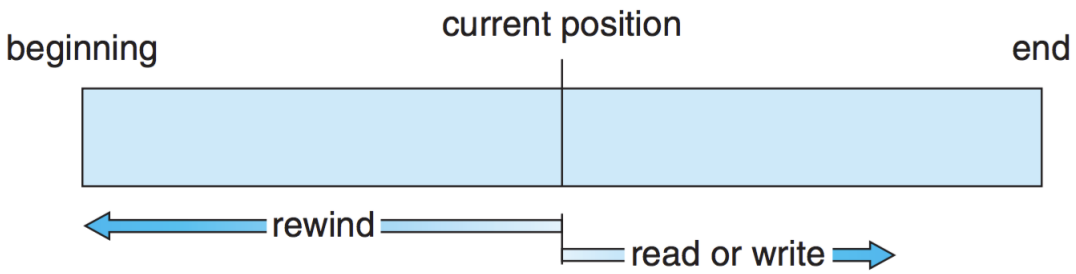


Figure 11.4 Sequential-access file.

현재 위치에서 읽거나 쓰면 offset이 자동으로 증가하고, 뒤로 돌아가기 위해선 되감기가 필요하다. 카세트 테이프 느낌이라 보면 된다.

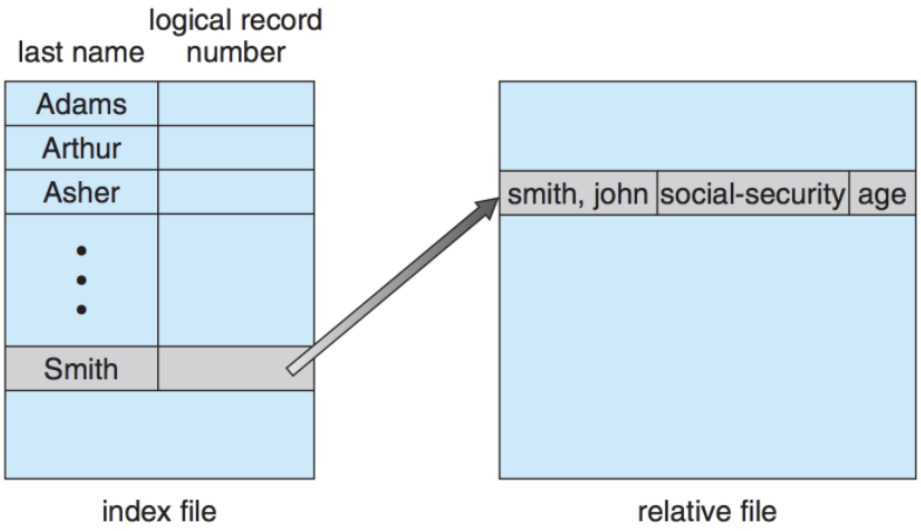
2. 직접 접근(Direct Access)

sequential access	implementation for direct access
reset	cp = 0;
read_next	read cp ; cp = cp + 1;
write_next	write cp; cp = cp + 1;

Figure 11.5 Simulation of sequential access on a direct-access file.

특별한 순서없이 빠르게 레코드를 read, write 가능하다. 현재 위치를 가리키는 cp 변수만 유지하면 직접 접근 파일을 가지고 순차 파일 기능을 쉽게 구현이 가능하다. 무작위 파일 블록에 대한 접근도 가능하기 때문에 순서의 제약이 없다. 대규모 정보를 접근할 때 유용하기 때문에 데이터 베이스에 활용된다.

3. 색인 접근(Index 접근)



파일의 레코드를 찾기 위해 인덱스를 먼저 찾고 대응되는 포인터를 얻는다. 이를 통해 파일에 직접 접근하여 원하는 데이터를 얻을 수 있다. 따라서 크기가 큰 파일에서 유용하다.

디렉토리

> 디렉토리(Directory)는 파일의 메타데이터 중 일부를 보관하고 있는 일종의 특별한파일이다.

해당 디렉토리에 속한 파일 이름과 속성들을 포함하고 있고, 파일 찾기, 생성, 삭제, 디렉토리 나열, 파일 시스템 순회 등의 기능을 제공한다.

디렉토리는 기본적으로 디렉토리의 파일을 빠르게 탐색할 수 있어야 한다. 또한 적절한 이름으로 사용자들이 편리하게 사용할 수 있으면 좋을 것이다. 그리고 파일들을 적절한 분류로 그룹화 해두면 사용하기 편리할 것이다.

1. 1단계 디렉토리(Single-Level Directory)

> 가장 간단한 구조

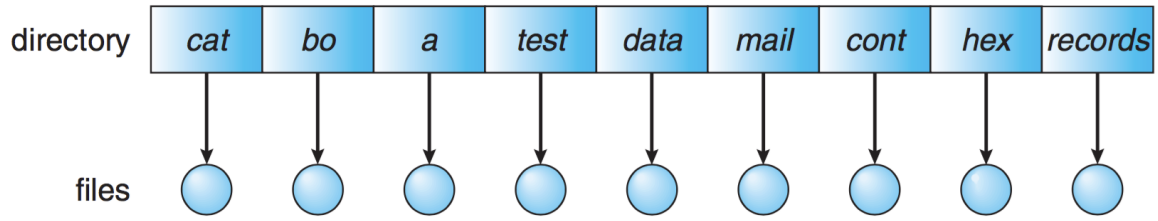


Figure 11.9 Single-level directory.

1단계 디렉토리는 **모든 파일들이 디렉토리 밑에 존재하는 형태**이다. 파일들은 서로 유일한 이름을 가지고 서로 다른 사용자라도 같은 이름을 사용할 수 없다. 지원하기도 쉽고 이해하기도 쉽지만, 파일이 많아지거나 다수의 사용자가 사용하는 시스템에선 심각한 제약을 가진다.

2. 2단계 디렉토리(Two-Level Directory)

> 사용자에게 개별적인 디렉토리를 만들어 줌

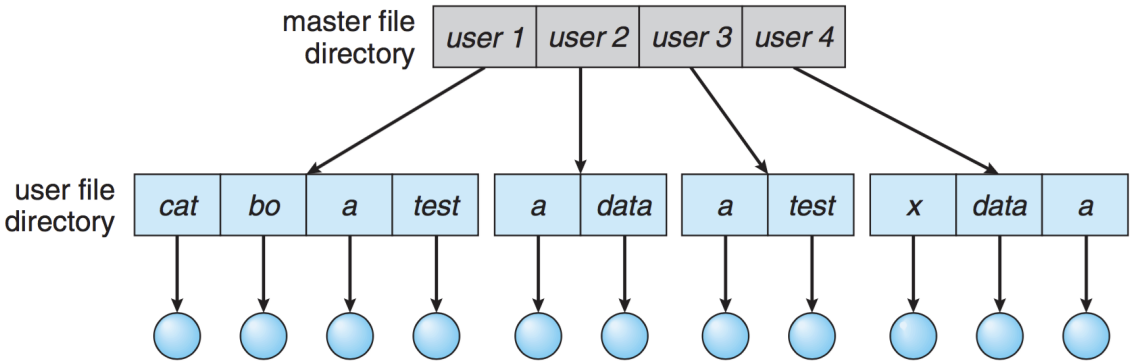


Figure 11.10 Two-level directory structure.

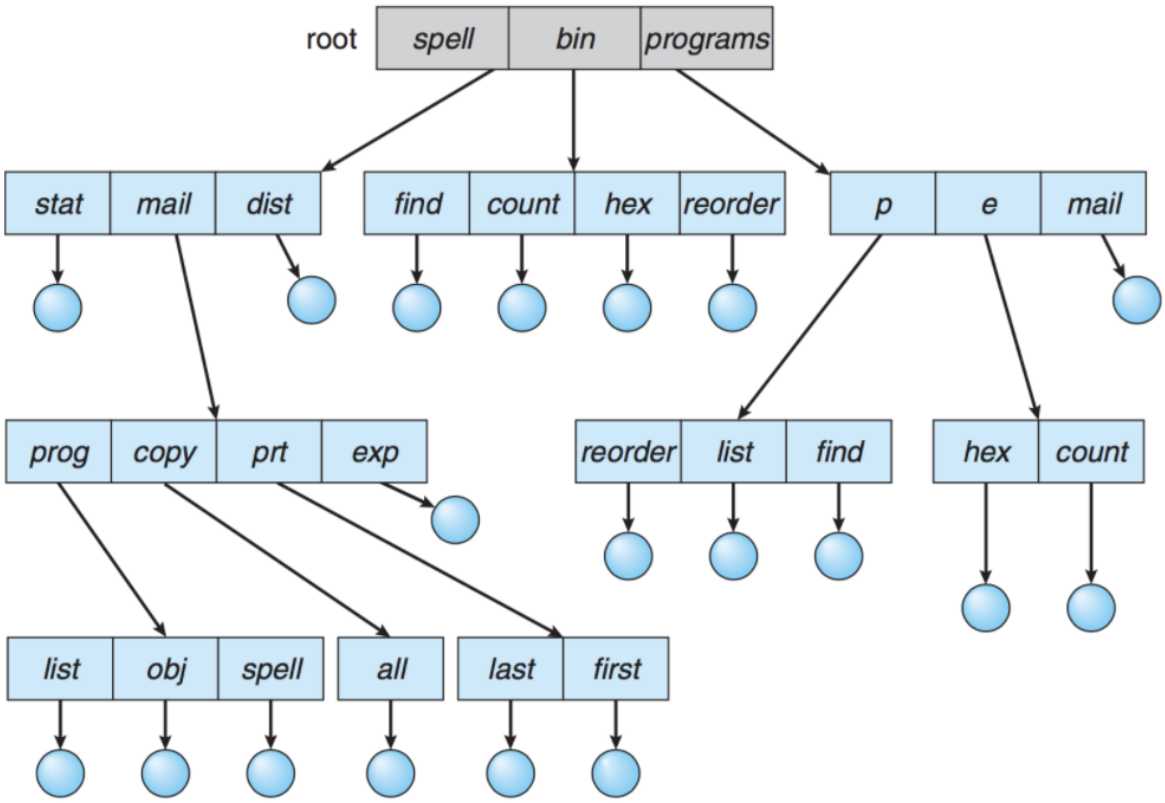
2단계 디렉토리는 **각 사용자별로 별도의 디렉토리를 갖는 형태**이다.

- UFD : 자신만의 사용자 파일 디렉토리
- MFD : 사용자의 이름과 계정번호로 인덱스 되어 있는 디렉토리이다. 각 엔트리는 사용자의 UFD를 가리킨다.

서로 다른 사용자가 같은 이름의 파일을 가질 수 있고 효율적인 탐색이 가능하다. 하지만 그룹화가 불가능하고 다른 사용자의 파일에 접근 하는 경우에 단점을 가진다.

3. 트리 구조 디렉토리(Tree-Structured Directory)

> 2단계 구조 확장된 다단계 트리 구조



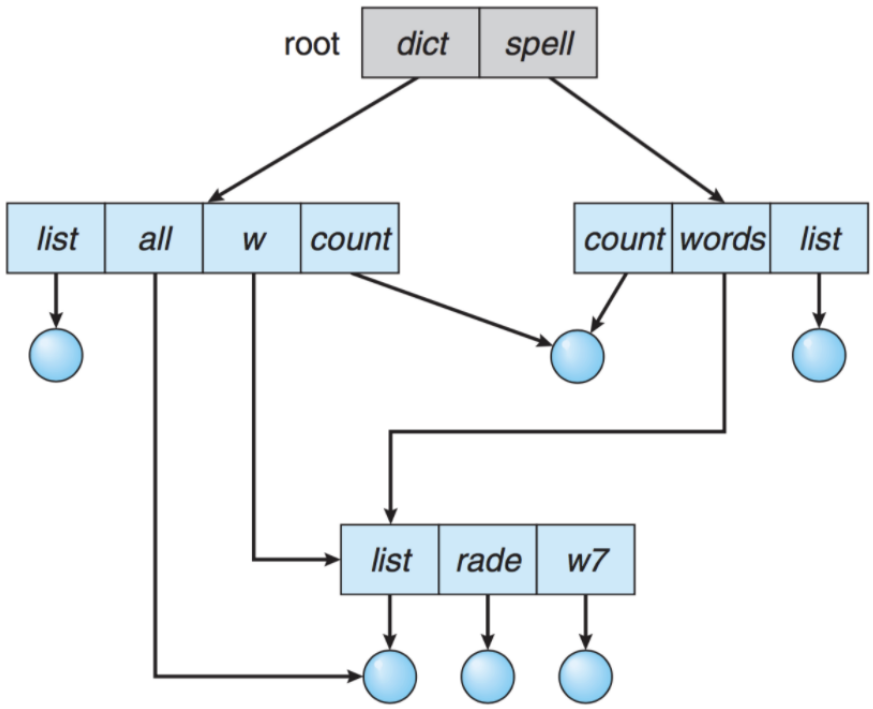
사용자들이 자신의 서브 디렉토리(Sub-Directory)를 만들어서 파일을 구성할 수 있다. 하나의 루트 디렉토리를 가지며 모든 파일은 고유한 경로(절대 경로. 상대 경로)를 가진다. 이를 통해 효율적인 탐색이 가능하고 그룹화가 가능하다.

디렉토리도 일종의 파일이므로 일반 파일과 디렉토리를 구분하기 위해 bit 하나를 사용하여 0이면 일반 파일, 1이면 디렉토리로 구분한다.

4. 비순환 그래프 디렉토리 (Acyclic-Graph Directory)

디렉토리들이 서브 디렉토리들과 파일을 공유할 수 있도록 한다. 트리 구조의 디렉토리를 일반화한 형태이다.

파일을 무작정 삭제하게 되면 현재 파일을 가리키는 포인터는 대상이 사라지게 된다. 따라서 참조되는 파일에 참조 계수를 두어서, 참조계수가 0이되면 파일을 참조하는 링크가 존재하지 않는 다는 의미이므로 그때 파일을 삭제하도록 한다.



5. 일반 그래프 디렉토리(General Graph Directory)

순환을 허용하는 그래프 구조이다. 순환이 허용되면 무한 루프에 빠질 수 있다.

따라서, 하위 디렉터리가 아닌 파일에 대한 링크만 허용하거나 garbage collection을 통해 전체 파일 시스템을 순회하고, 접근 가능한 모든 것을 표시한다.