파일 시스템

File

File

- 컴퓨터에서 의미가 있는 정보를 담은 논리적인 단위 → 관련된 정보 자료들의 집합
- 레코드 또는 블록 단위로 하드디스크 에 저장
- 운영체제는 다양한 저장 장치를 file 이라는 동일한 논리적 단위로 볼 수 있게 해준다
- 실행 파일과 데이터 파일로 존재
    - **실행 파일** : 운영체제가 메모리로 가져와 CPU를 이용하여 작업하는 파일 (ex 윈도우 exe 파일. 유닉스는 따로 확장자 없음)
    - **데이터 파일** : 실행파일이 작업하는데 필요한 데이터를 모아놓은 파일

File attribute (metadata)

- 파일 자체의 내용이 아니라 파일을 관리하기 위한 각종 정보들
- 파일 이름, 유형, 저장된 위치, 접근 권한, 소유자 등에 대한 정보

Directory

- 파일의 메타데이터 중 일부를 보관하고 있는 일종의 특별한파일
- 자신의 디렉토리에 속한 파일 이름 및 메타데이터 정보를 담음

File System

- **정의**
    - 컴퓨터에서 파일이나 자료를 쉽게 발견 할 수 있도록, 유지, 관리하는 방법.
    - 파일 및 메타데이터, 디렉토리 정보 등을 관리
- **특징**
    - 사용자 영역이 아닌 커널 영역에서 동작
    - 파일을 빠르게 읽기, 쓰기, 삭제 등 기본적인 기능을 원활히 수행하기 위한 목적
    - 계층적 디렉터리 구조를 가짐

    - 디스크 파티션 별로 하나씩 둔다
- **목적**
    - 파일 관리를 용이하게 한다.
    - 하드디스크의 막대한 용량을 효율적으로 이용
    - 하드디스크와 메인 메모리 속도차 줄이기 : 하드디스크와에 저장된 데이터들은 하드디스크와에서 실행되는 것이 아니라 메인 메모리에 로드 되어 사용된다. 이 때 파일 시스템이 하드디스크와에 저장된 데이터들의 목차가 되어 데이터를 실행하려고 클릭 했을 때 메인 메모리에 빠르게 로드 될 수 있도록 한다.
- **역할**
    - 불의의 사태를 대비하여 파일의 예비(Backup)와 복구(Recovery) 등의 기능을 제공한다.
    - 사용자가 파일을 편리하게 사용할 수 있도록 파일의 논리적 상태(디렉터리)를 보여주어야 한다.
    - 파일을 안전하게 사용할 수 있도록 하고, 파일이 보호되어야 한다.
    - 파일의 정보가 손실되지 않도록 데이터 무결성을 유지해야 한다.

File Descriptor

- 프로세스는 FD(file descriptor)를 통해 파일에 접근한다.
- **시스템으로부터 할당 받은 파일을 대표하는 0이 아닌 정수 값**
- 프로세스에서 열린 파일의 목록을 관리하는 테이블의 **인덱스**
- 프로세스가 실행 중 파일을 Open하면 커널은 해당 프로세스의 파일 디스크립터 숫자 중 사용하지 않는 가장 작은 값을 할당해준다. (프로세스 내에서만 고유한 값)
- 그 다음 프로세스가 열려있는 파일에 시스템 콜을 이용해서 접근할 때, 파일 디스크립터(FD)값을 이용해서 파일을 지칭할 수 있다.

Directory와 디스크 구조

1) 1단계(단일) 디렉터리

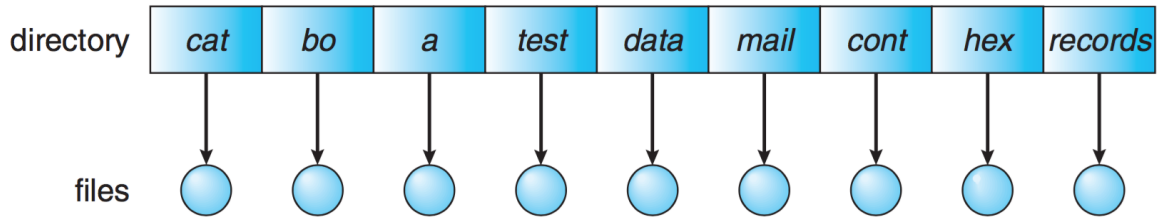


Figure 11.9 Single-level directory.

- 모든 파일이 같은 디렉터리에서 관리되는 가장 간단한 구조
- 모든 파일이 유일한 이름을 가져야 함
- 다수의 사용자가 사용하거나 파일 수가 증가할 때 관리가 어려움

2) 2단계 디렉터리

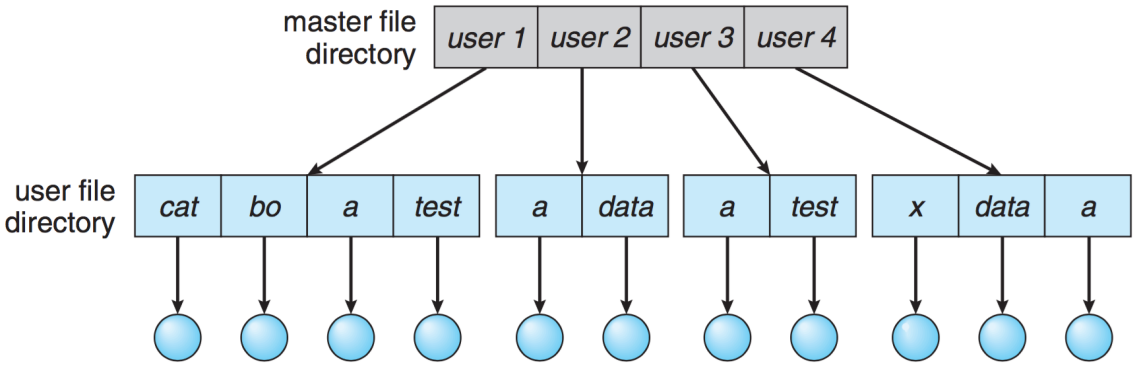


Figure 11.10 Two-level directory structure.

- 중앙에 **마스터 파일 디렉터리(MFD : Master File Directory)** 가 있고, 그 아래에 **각 사용자에게 할당하는 디렉터리(UFD : User File Directory)** 가 있는 구조
- 각 사용자에게 서로 다른 디렉터리를 할당
- 사용자는 각자 다른 디렉터리를 가지므로 다른 사용자와 같은 이름의 파일 소지 가능
- 각 사용자별 디렉터리끼린 독립적이므로 파일을 공유할 수 없음

3) 트리 구조 디렉터리

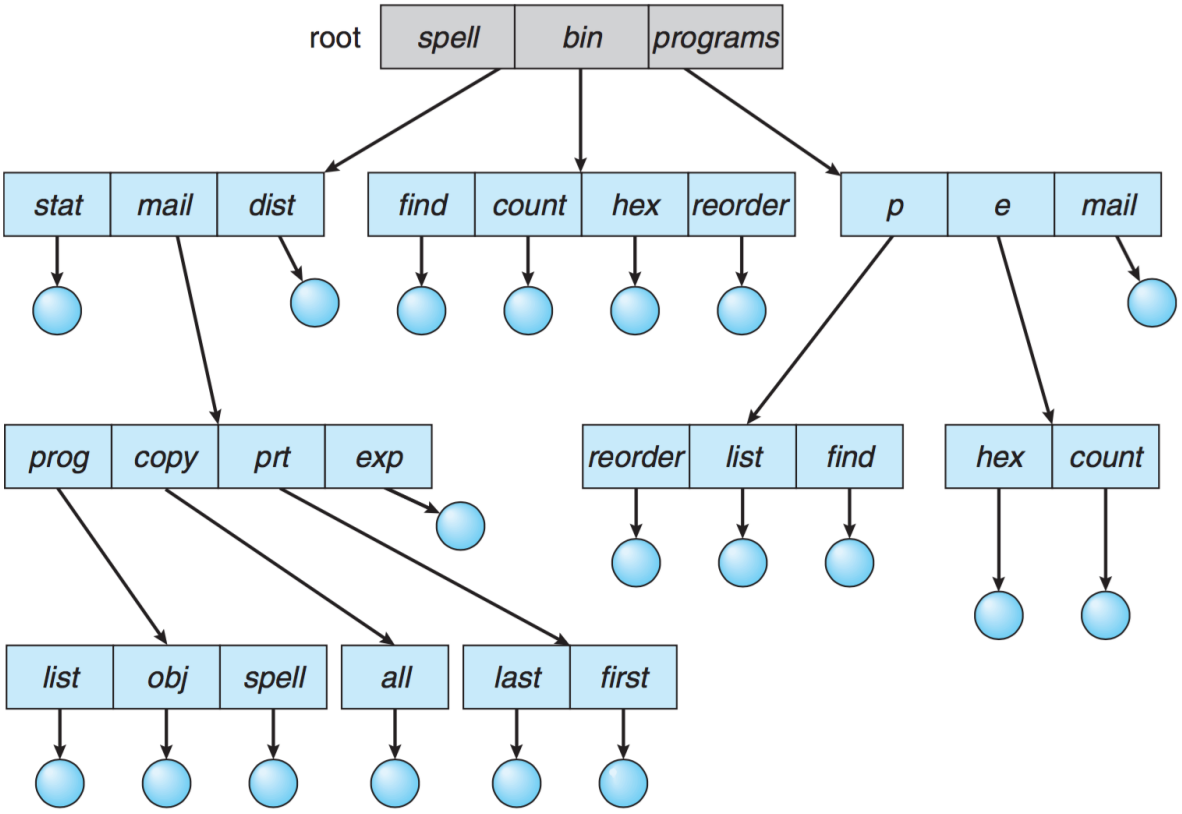
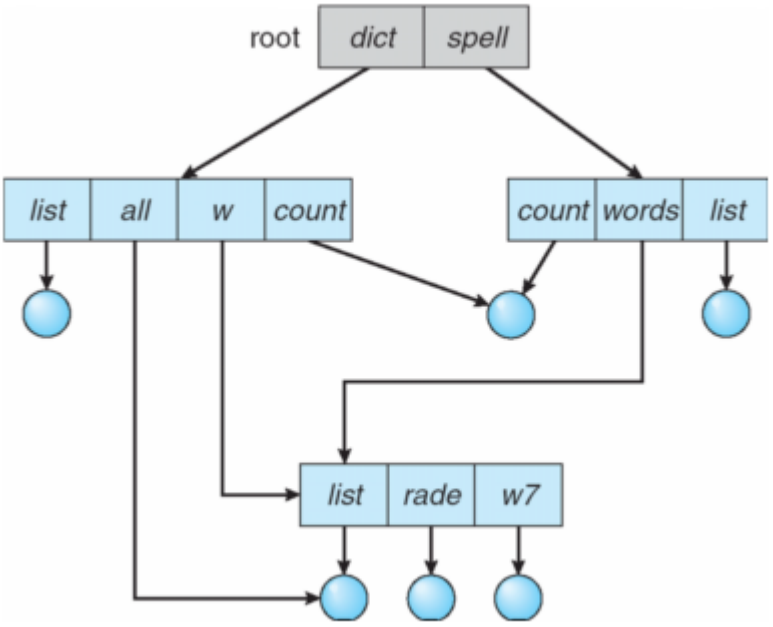


Figure 11.11 Tree-structured directory structure.

- 2단계 구조 확장된 다단계 트리 구조를 지닌 디렉터리
- 하나의 루트 디렉터리와 다수의 서브 디렉터리로 구성
- 서로 다른 디렉터리 내에서는 동일 명칭의 파일이나 디렉터리 생성 가능
- 파일 및 디렉터리 탐색은 **절대경로 혹은 상대경로**를 이용
  - **절대경로** : 루트 디렉터리를 기준으로 해당 파일 혹은 종속 디렉터리에 이르는 경로
  - **상대경로** : 현재 디렉터리를 기준으로 해당 파일 혹은 디렉터리까지 경로
- Dos, windows, Unix 운영체제에서 사용하는 구조

4) 비순환그래프 디렉토리(Acyclic Graph Directory)



- 하위 파일 또는 하위 디렉터리를 공동으로 사용할 수 있으며, 사이클이 허용되지 않는 구조
- 하나의 파일이나 디렉터리가 여러 개의 경로 이름을 가질 수 있음
- 디렉터리 구조가 복잡하고, 공유된 하나의 파일을 탐색할 경우 다른 경로로 두 번 이상 찾아갈 수 있으므로 시스템 성능이 저하될 수 있음

4) 그래프 디렉터리

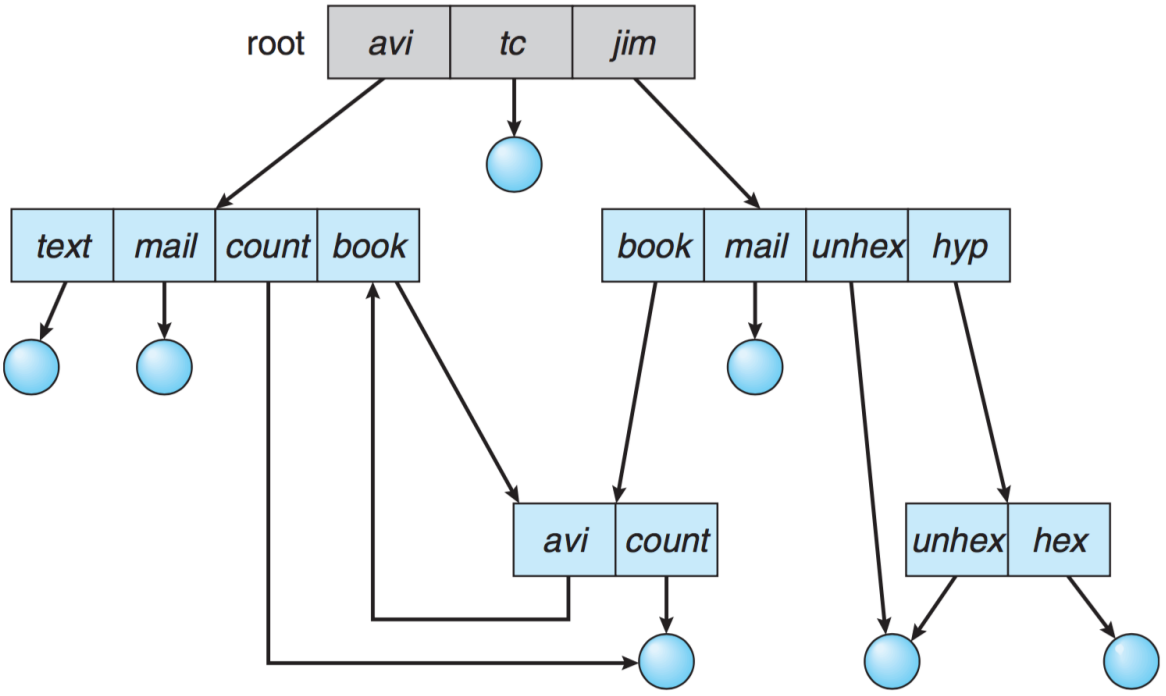


Figure 11.13 General graph directory.

- 트리 구조에 링크를 추가하여 사이클이 허용되는 구조
- 가비지 컬렉션을 이용해 전체 파일 시스템을 순회하고 접근 가능한 모든 것을 표시
- 불필요한 파일 제거를 위해 참조 계수기(countert) 필요
- 제거된 파일의 디스크 공간 확보를 위해 쓰레기 수집 필요

파일 접근

1) 순차 접근

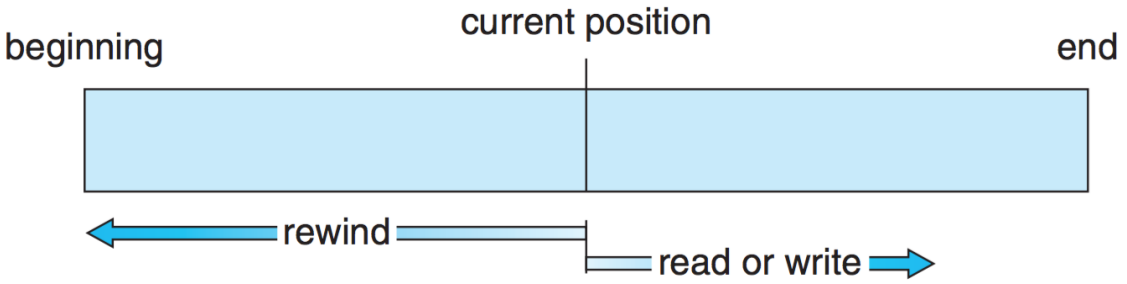


Figure 11.4 Sequential-access file.

- 가장 간단한 방법으로 파일의 정보가 레코드 순서대로 처리
- 대부분 연산은 read와 write
- 현재 위치를 가리키는 포인터에서 시스템 콜이 발생할 경우 포인터를 앞으로 보내면서 read와 write를 진행. 뒤로 돌아갈 땐 지정한 offset만큼 되감기를 해야 한다. (테이프 모델 기반)

2) 직접 접근

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 가능
- 현재 위치를 가리키는 변수 cp만 유지된다면 직접 접근 파일을 가지고 순차 파일 기능을 쉽게 구현할 수 있음
- 대규모 정보를 즉각적으로 접근하는 데 유용하여 데이터베이스에 이용됨

3) 색인 접근

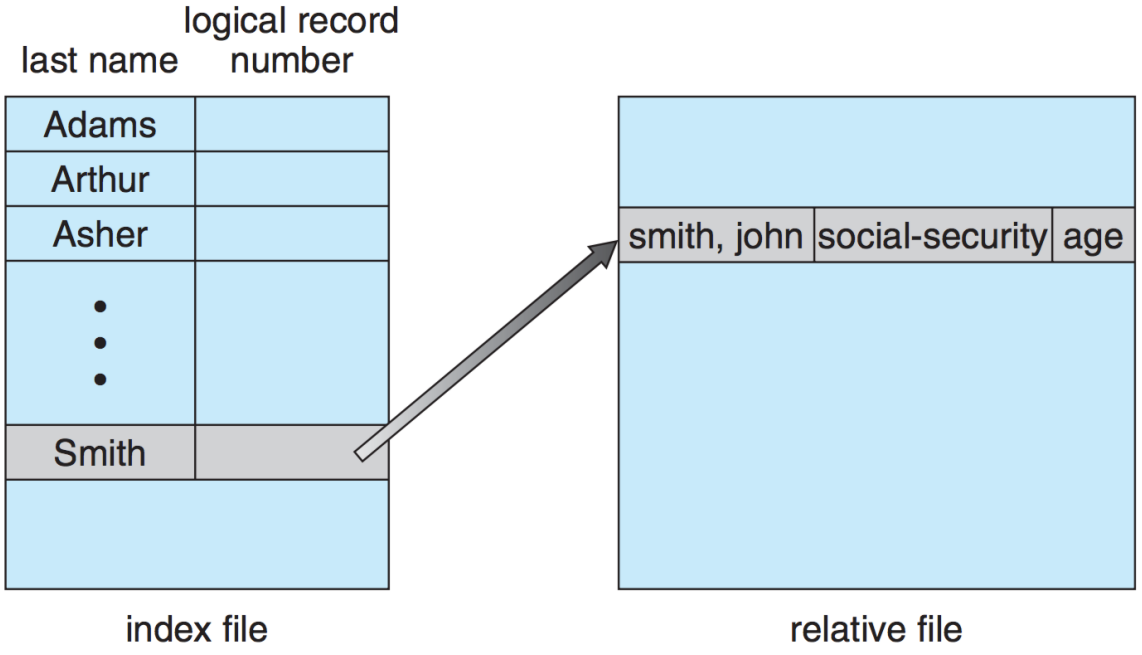


Figure 11.6 Example of index and relative files.

- 직접 접근 파일에 기반하여 색인 구축
- 크기가 큰 파일을 입출력 탐색할 수 있게 도와줌