

File System

≡ 태그	OS
≡ 주차	6주차



목차

0. File

1. File System

1-1. File System

- 1) 특징
- 2) 역할
- 3) 개발 목적
- 4) 종류

1-2. Access Methods (접근 방법)

- 1) Sequential Access (순차 접근)
- 2) Random Access (임의 접근)
- 3) Index Access (색인 접근)

1-3. File Allocation (디스크 할당)

- 1) Contiguous Allocation (연속 할당)
- 2) Linked List Allocation (연결 할당)
- 3) Index Allocation (인덱스 할당)
- 4) I-nodes (아이노드)

2. Directory Structure

2-1. Directory Structure

- 1) 기능
- 2) 구성

2-2. Directory Level

- 1) Single Level Directory
- 2) Two Level Directory
- 3) Tree Structured Directories
- 4) Acyclic Graph(비순환 그래프) Directory
- 5) General Graph(일반 그래프) Directory

0. File

- 논리적인 저장 단위
- 관련된 정보 자료들의 집합에 이름붙인 것
- 일반적으로 레코드(Record) 혹은 블록(Block) 단위로 비휘발성 보조기억장치에 저장 됨
- 구조
 - **MetaData:** 데이터 영역에 기록된 파일의 이름, 위치, 크기, 시간정보, 삭제유무 등의 파일 정보
 - **데이터 영역:** 파일의 데이터



File Attribute (MetaData)

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

1. File System

1-1. File System

- 컴퓨터에서 파일이나 자료를 쉽게 발견할 수 있도록, 유지 및 관리하는 방법
- 저장매체에 있는 수많은 파일들을 관리하는 방법

1) 특징

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



파티션 (Partition)

- 연속된 저장 공간을 하나 이상의 연속되고 독립적인 영역으로 나누어 사용할 수 있도록 정의한 규약
- 하나의 물리적 대스크 안에 여러 파티션을 두는 것이 일반적
- 여러 물리적 디스크를 하나의 파티션으로 구성하기도 함

2) 역할

- 파일 관리
 - Create, Write, Read, Delete, Reposition, Truncate(파일 내부 내용 삭제)
- 파일 무결성 매커니즘
 - 파일 원본이 조작되었는지 확인하는 것
- 보조 저장소 관리
- 접근 방법 제공

3) 개발 목적

- 하드디스크와 메인 메모리 사이의 속도 차이를 줄이기 위함
- 파일 관리
- 하드디스크의 용량을 효율적으로 이용하기 위함

4) 종류

- FAT 16/32 : MS-DOS
- NTFS : Window NT 등
- UFS : UNIX

- EXT 1~4 : POSIX(LINUX 확장 파일 시스템)
 - 리눅스 계열은 실험적인 것을 포함해 40개 이상의 파일 시스템 지원
- CDFS : CD 저장매체(CD-ROM)(ISO 9660 표준)
- VFAT : USB용
- NFS : 원격 저장소

1-2. Access Methods (접근 방법)

1) Sequential Access (순차 접근)

- 가장 단순한 접근 방법으로 파일의 정보가 레코드 순서대로 처리
- 카세트 테이프를 사용하는 방식과 동일
 - 현재 위치에서 읽거나 쓰면 offset이 자동으로 증가
 - 뒤로 돌아가기 위해서는 되감기 필요
- 요즘에는 사용 X



제공하는 연산

- read next: 오직 앞의 방향으로만 작동
- write next: 오직 앞의 방향으로만 작동
- reset: Pointer 초기화

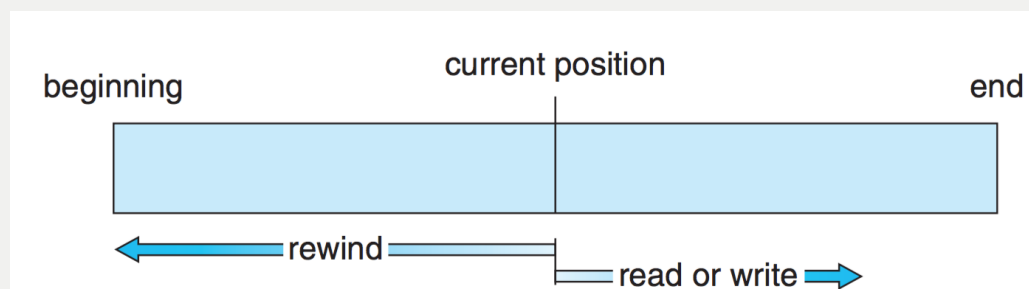


Figure 11.4 Sequential-access file.

2) Random Access (임의 접근)

- 파일의 레코드를 임의의 순서로 접근할 수 있음
- 순서에 상관 없이 빠르게 레코드를 읽고 쓸 수 있음
- 현재 위치를 가리키는 cp 변수만 유지하면 쉽게 순차접근 구현 가능
- 대규모 정보를 접근할 때 유용하기 때문에 DB에 활용



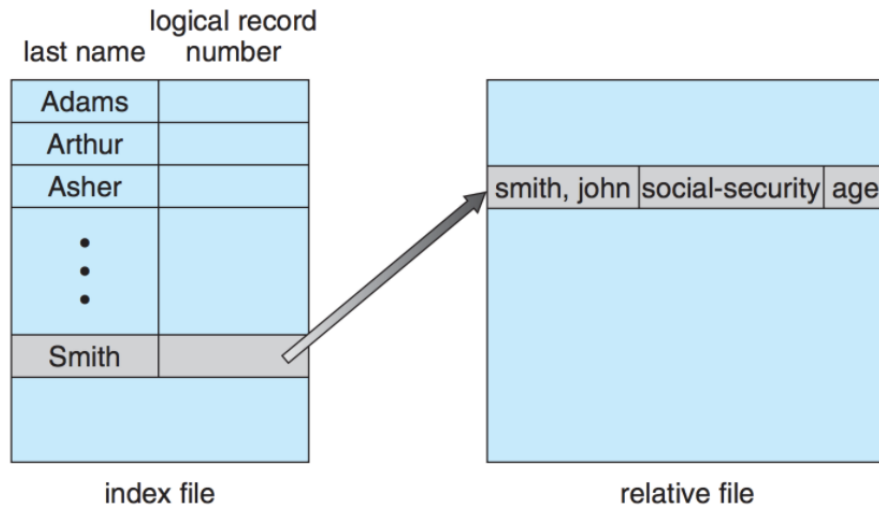
제공하는 연산

- read n: n번째 블록에 접근
- write n: n번째 블록에 접근
- position to n: n번째 블록 위치로 Pointer 이동
- rewrite n

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.

3) Index Access (색인 접근)



- 파일에서 레코드를 찾기 위해 색인을 먼저 찾고 대응되는 포인터를 얻음
- 이를 통해 파일에 직접 접근하여 원하는 데이터를 얻을 수 있음
- 크기가 큰 파일을 입출력 탐색할 수 있게 도와주는 방법

1-3. File Allocation (디스크 할당)

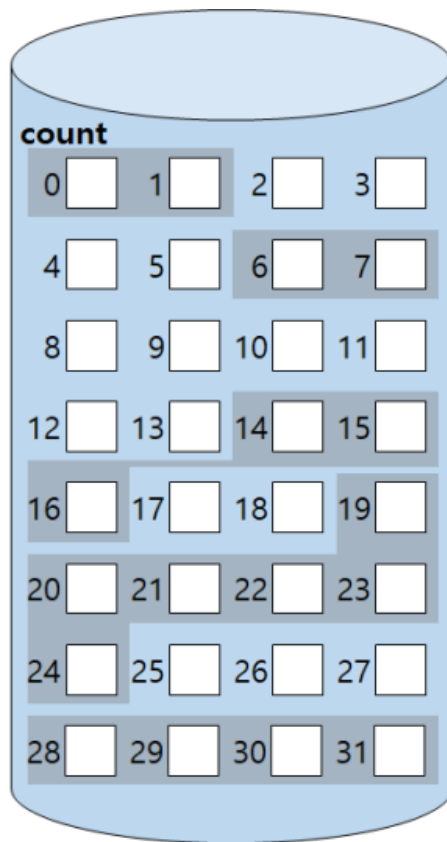
- File의 내용을 담고 있는 Data Block의 위치 정보를 어떻게 저장할지 고려해야 함
- OS마다 각기 다른 방식으로 File의 Data Block을 관리함



Disk Block

File System의 입출력 단위

1) Contiguous Allocation (연속 할당)

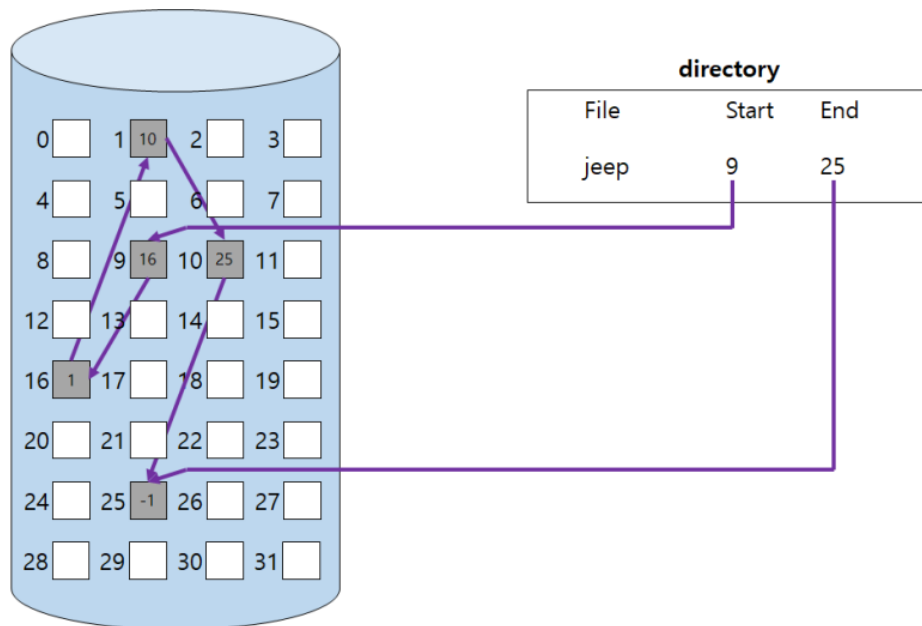


directory

File	Start	Length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	4

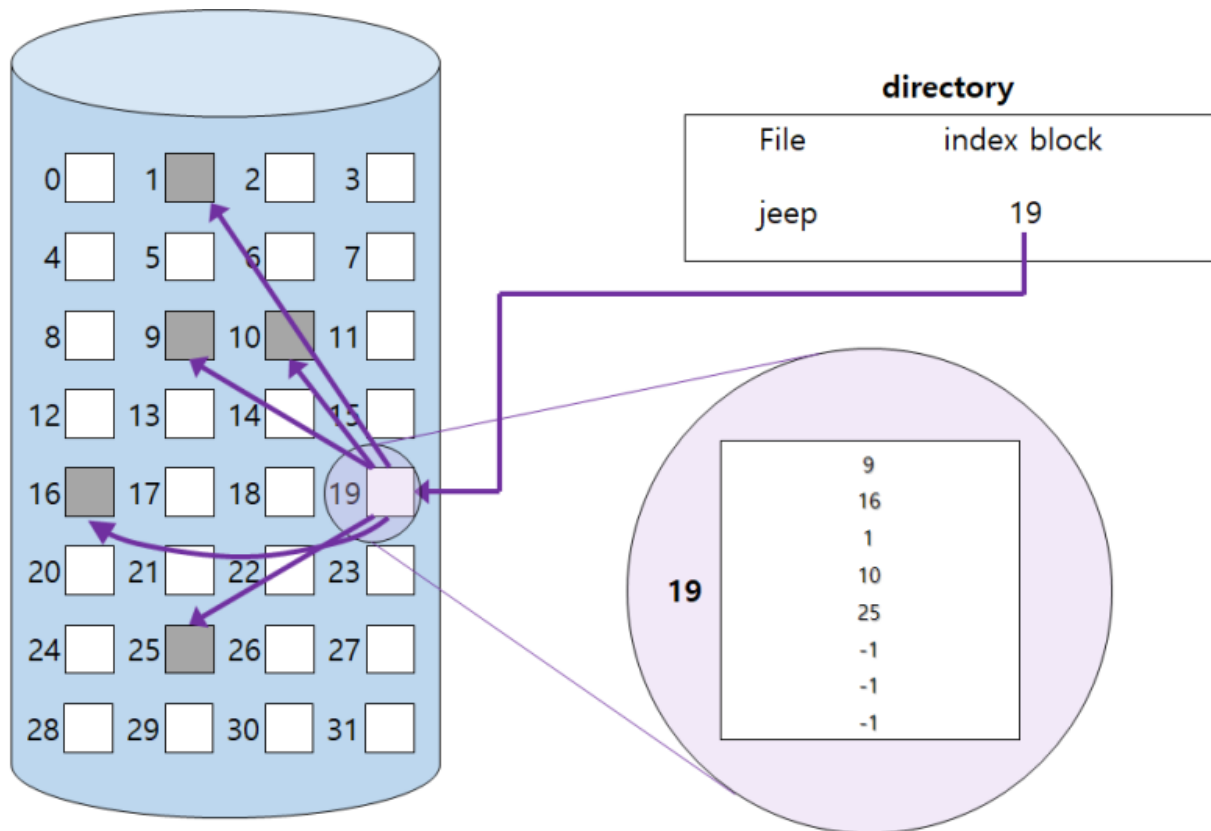
- File을 물리적으로 연속된 Disk Block에 저장
- 장점
 - 구현이 간단함
 - 물리적으로 연속된 공간에 있으므로 전체 File을 한번에 읽어들이기 경우 성능이 매우 뛰어남
- 단점
 - File은 반드시 한번에 끝까지 기록되어야 함
 - OS에서 File의 끝에 예비용 Block을 남겨둔 경우 Disk 공간 낭비 (Fragmentation 심화)

2) Linked List Allocation (연결 할당)



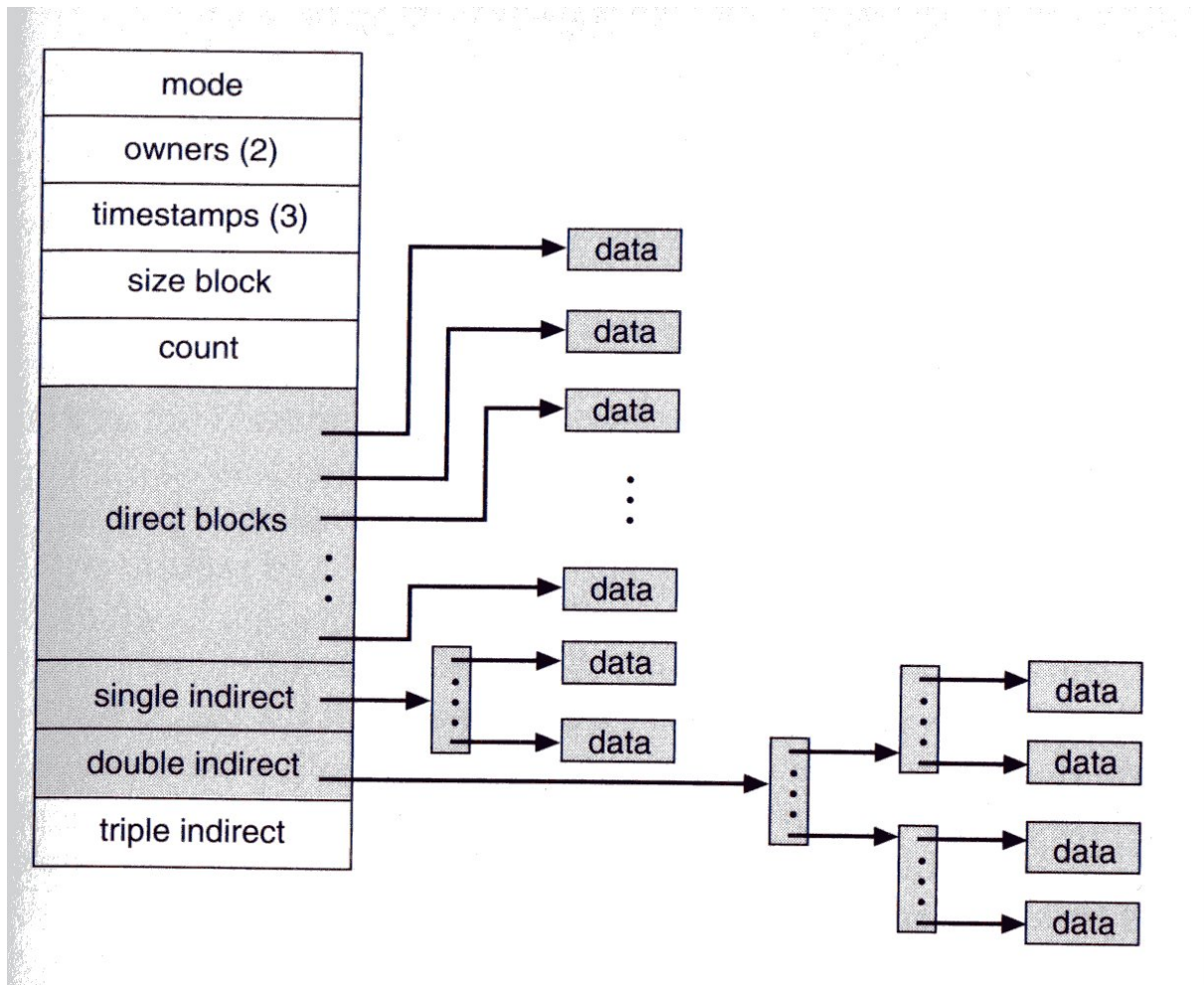
- Disk의 Block을 Linked List로 구현하여 데이터 저장
- 장점
 - File의 Data Block은 Disk의 어디든지 위치할 수 있음
 - 공간의 낭비가 없음
- 단점
 - Random Access 불가능 (인덱스 탐색 불가)
 - File의 특정 위치를 찾기 위해 시작 node부터 탐색해야 함
 - 다음 Data Block에 대한 Pointer로 인해 Data Block에서 Data를 저장하는 공간이 반드시 2의 배수가 아닐 수도 있음
 - 대부분의 프로그램의 Read/Write 단위가 2의 배수이기 때문에 성능 측면에서 뒤쳐질 수 있음

3) Index Allocation (인덱스 할당)



- File의 Data와 관련된 Block을 하나의 Block에 모아둠
 - File의 Data Block 중 하나를 Index Block이라 하여, 모든 Data Block의 위치를 Index Block에서 알 수 있음
- Random Access가 가능해 Linked List Allocation에 비해 빠름
- 단점
 - 최대 File의 크기가 고정됨
 - Index Block의 크기가 고정되어있어 가용한 Pointer의 수가 한정됨

4) I-nodes (아이노드)



- File에 대한 Data Block Index들을 Table 형태로 관리하는 방법
- 구성
 - Field: File에 대한 속성 표시
 - Direct Index: 작은 크기의 File들을 위한 인덱스
 - Index Table: File의 크기가 커짐에 따라 요구되는 Data Block의 Index를 저장하기 위한 테이블
 - Single Indirect Block
 - Double Indirect Block
 - Triple Indirect Block



MS-DOS: Linked List Allocation

UNIX: I-node

2. Directory Structure

- Directory는 파일의 메타데이터 중 일부를 보관하고 있는 일종의 특별한 파일
- 해당 Directory가 속한 파일 이름과 속성을 포함

2-1. Directory Structure

1) 기능

- Create (파일 생성)
- Search (파일 찾기)
- Rename (파일 이름 수정)
- Delete (파일 삭제)
- List (Directory 나열)
- Traverse (파일 시스템 순회)

2) 구성

- Efficiency: Directory는 파일을 빠르게 탐색할 수 있어야 함
- Naming: 사용하기 편리한 적절한 이름 사용
- Grouping: 파일들을 적절한 분류로 그룹화

2-2. Directory Level

1) Single Level Directory

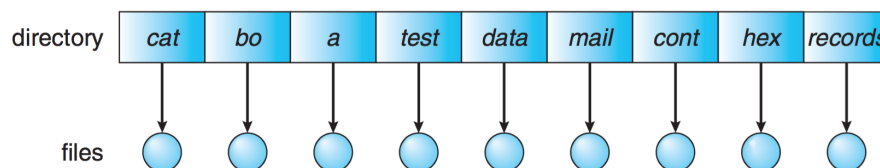


Figure 11.9 Single-level directory.

- 모든 파일들이 **Directory** 밑에 존재하는 형태
- 가장 간단한 구조

- 파일들은 서로 유일한 이름을 가짐
 - 서로 다른 사용자라도 같은 이름의 파일을 사용할 수 없음
- 파일이 많아지거나 다수의 사용자가 사용하는 시스템이라면 문제 발생
 - Naming Problem
 - Grouping Problem: 시스템에 오직 1개의 Directory만 있어 Grouping 불가능

2) Two Level Directory

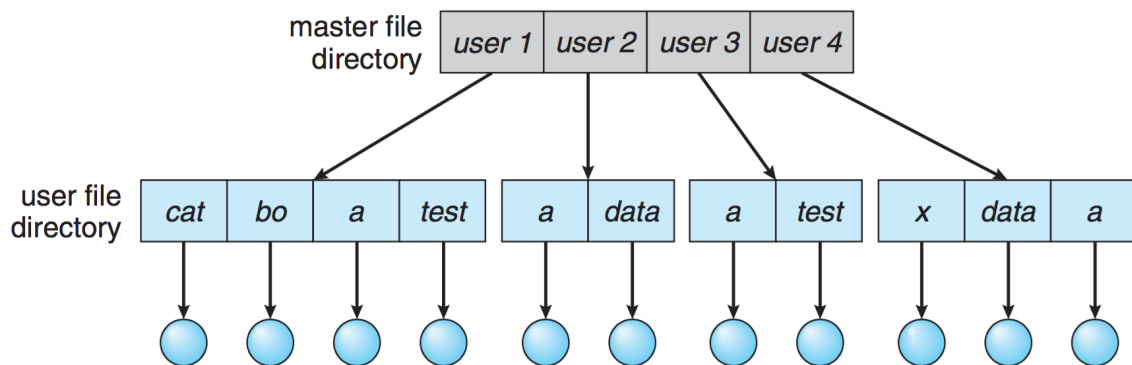


Figure 11.10 Two-level directory structure.

- **각 사용자별로 별도의 Directory를 가짐**
- 서로 다른 사용자간에는 파일 이름 중복 허용
- 효율적인 탐색 가능
 - 각 파일들은 경로명으로 찾아가야 함
- 종류
 - **UFD(User File Directory):** 자신만의 사용자 파일 디렉토리
 - **MFD(Master File Directory):** 사용자의 이름과 계정 번호로 색인되어 있는 Directory. 각 엔트리는 사용자의 UFD를 가르킴
- Grouping 기능이 존재하지 않음

3) Tree Structured Directories

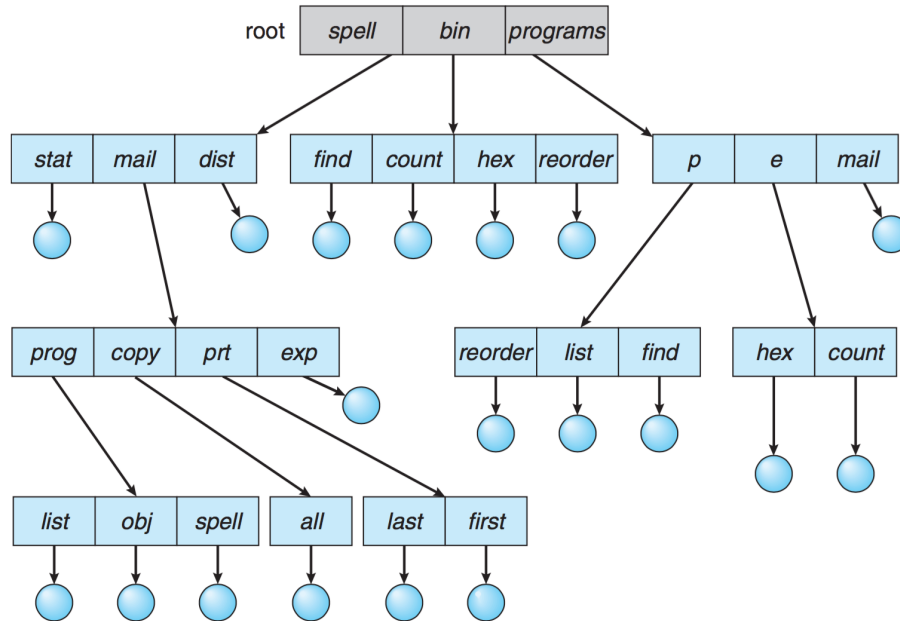
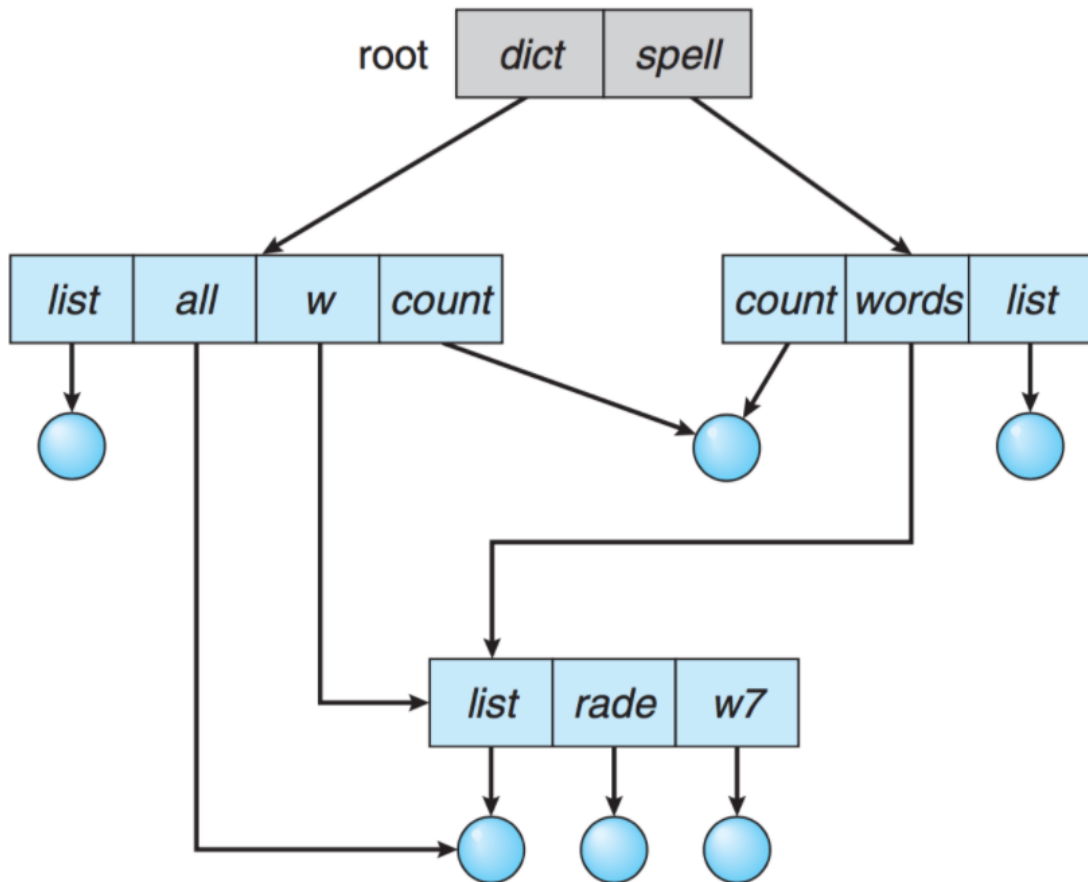


Figure 11.11 Tree-structured directory structure.

- 사용자들이 자신의 서브 디렉토리(Sub Directory)를 만들어 파일 구성 가능
- 각 사용자는 하나의 루트 디렉토리를 가지면, 모든 파일은 고유한 경로(절대/상대 경로)를 가짐
 - **절대 경로(Absolute path)**: 파일을 읽기 위한 파일 전체 경로 (root부터 현재 파일 위치까지 포함)
 - **상대 경로(Relative Path)**: 현재 위치를 기준으로 하여 목적지까지의 상대적인 경로
- 해당 경로를 통해 효율적인 탐색 및 그룹화 가능
- 디렉토리는 일종의 파일이므로, 일반 파일인지 Directory인지 구분할 필요가 있음
 - 이를 bit를 사용해 구분
 - 0: 일반 파일
 - 1: 디렉토리

4) Acyclic Graph(비순환 그래프) Directory



- Directory들이 Sub Directory들과 파일을 공유할 수 있도록 함
- 서로 다른 사용자가 동일한 파일에 대해서 서로 다른 Alias(별칭) 사용 가능
- 파일을 무작정 삭제하면 현재 파일을 가리키는 포인터는 대상이 사라지게 됨
 - Dangling Pointer 발생
- 따라서, 참조되는 파일에 참조 계수 존재
 - 참조 계수가 0이 되면 파일을 참조하는 링크가 존재하지 않는다는 의미이므로, 그 때 파일을 삭제할 수 있도록 함

5) General Graph(일반 그래프) Directory

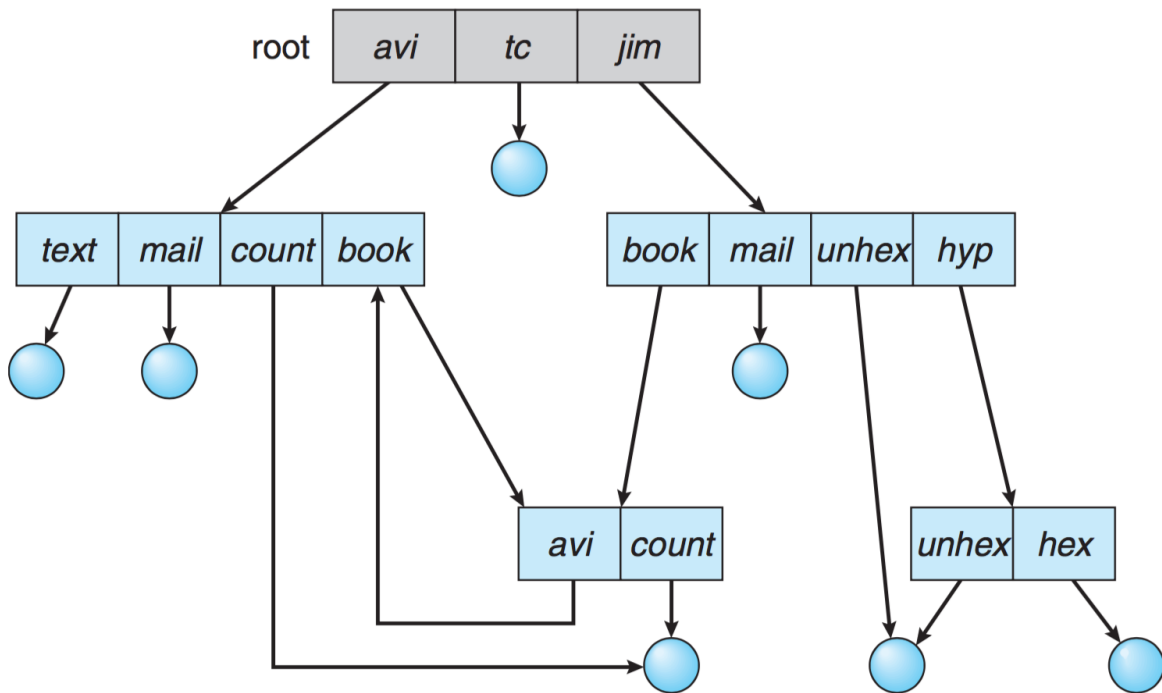



Figure 11.13 General graph directory.

- 순환을 허용하는 그래프 구조
- 무한 루프에 빠질 수 있어 잘 쓰이지 않음
- 순환이 발생하지 않도록 하위 디렉토리가 아닌 파일에 대한 링크만 허용하거나, Garbage Collection을 통해 전체 파일 시스템을 순회하고, 접근 가능한 모든 것을 표시함

출처

File System (파일 시스템)

정의논리적인 저장 단위로, 관련된 정보 자료들의 집합에 이름을 붙인 것이다. 일반적으로 레코드 (Record) 혹은 블록(Block)단위로 비휘발성 보조기

 <https://velog.io/@yuseogi0218/File-System-파일-시스템>

File System