쉽게 배우는

운영체제

2판

Chapter 11 파일 시스템

차례

01 파일과 파일 시스템

02 디렉터리의 구조

03 디스크 파일 할당

03 [심화학습] 유닉스 파일의 특징

학습목표

- 파일 시스템의 역할을 이해하고 파일의 속성, 작업 유형, 구조를 알아본다.
- 디렉터리의 개념과 구조를 살펴보고 디렉터리가 어떻게 운영되는지 이해한다.
- 디스크 파일 할당 방식과 빈 공간 관리 방법을 알아본다.

1-1 파일 시스템의 개요

■ 파일 시스템의 개념

- 파일을 보관/관리하는 파일 관리자를 두어 저장장치의 전체 관리를 맡기는 것
- 파일 관리자는 파일 테이블을 사용하여 파일 관리
- 사용자가 특정 파일에 접근하려면 파일 관리자로부터 파일에 접근할 수 있는 권한(키) 획득해야 함. 파일 접근 권한을 파일 디스크립터(file descriptor)라고 함

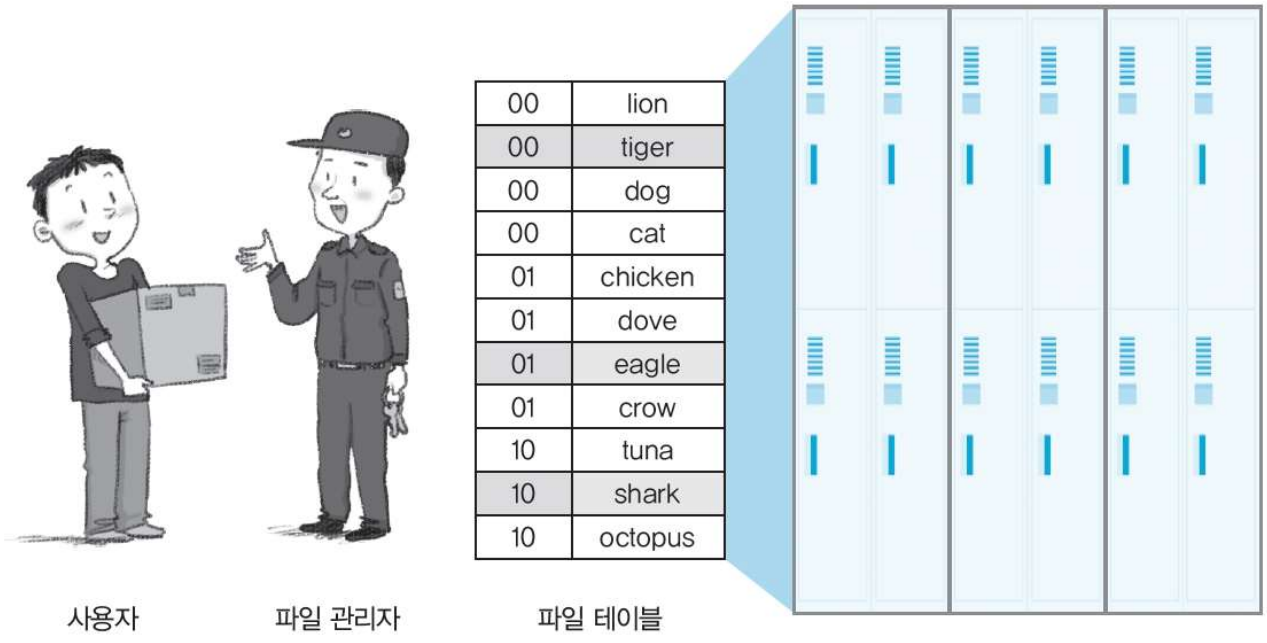

그림 11-1 사용자와 파일 관리자

1-1 파일 시스템의 개요

■ 파일 시스템의 기능

표 11-1 파일 시스템의 기능

기능	설명
파일 및 디렉터리 구성	사용자의 요구에 따라 파일과 디렉터리를 만든다.
파일 및 디렉터리 관리	파일 및 디렉터리의 생성, 수정, 삭제 등을 관리하며, 사용자가 파일 및 디렉터리에 접근할 수 있도록 한다.
접근 방법 제공	파일과 디렉터리에서 읽고 쓰고 실행할 수 있도록 사용자에게 접근 방법을 제공한다.
접근 권한 관리	다른 사용자에게서 파일 및 디렉터리를 보호하기 위해 접근 권한을 관리한다.
무결성 보장	파일 내용이 손상되지 않도록 무결성을 보장한다.
백업과 복구	파일을 보호하기 위해 백업과 복구 작업을 한다.
암호화	파일을 암호화하여 악의적인 접근으로부터 파일을 보호한다.

1-1 파일 시스템의 개요

■ 블록과 파일 테이블

- 블록(block): 저장장치에서 사용하는 가장 작은 단위, 한 블록에 주소 하나 할당
- 파일 테이블에는 각 파일이 어떤 블록에 저장되어 있는지에 대한 정보 저장

이름	블록 번호
파일 A	1, 3, 9
파일 B	4, 2
파일 C	13
파일 D	15, 12
파일 E	23, 7

파일 테이블

	0	1	2	3	4	5	6	7	8	9
0		A	B	A	B			E		A
1			D	C		D				
2				E						
3										
4										
5										

블록 번호

저장장치

그림 11-2 파일 테이블과 블록

1-2 파일

■ 파일 종류와 확장자

- 실행 파일: 운영체제가 메모리로 가져와 CPU를 이용하여 작업하는 파일
- 데이터 파일: 실행 파일이 작업하는 데 필요한 데이터를 모아놓은 파일

표 11-2 파일의 분류와 확장자

파일	확장자	설명
실행 파일	exe, com	유닉스에는 실행 파일 확장자가 없음
소스코드 파일	c, cpp, pas, a, java	다양한 소스코드의 확장자
라이브러리 파일	lib, a, dll	소스코드를 위한 라이브러리의 확장자
배치 파일	bat, sh, csh	초기 배치 파일의 확장자
문서 파일	txt, doc, hwp, pdf, ps 등	문서 데이터 파일의 확장자
동영상 파일	avi, asf, mkv, mov, rmv	동영상 데이터 파일의 확장자
음악 파일	wav, mp3, ogg, flc, aac	음악 데이터 파일의 확장자
이미지 파일	bmp, gif, jpg, png, webp	이미지 데이터 파일의 확장자
압축 파일	rar, zip, arc, al	압축 파일의 확장자

1-2 파일

■ 파일 이름

- ‘파일이름.확장자’ 형태로 구성
- 초창기 운영체제는 파일 이름 8자, 확장자 3자로 제한. 아직도 파일 확장자는 3자 이하 대부분이지만 4자나 5자로 된 확장자도 있음
- 파일 이름에 마침표(.) 여러 번 사용 가능. 마지막 마침표 다음 글자는 확장자로 인식
- 파일 이름은 현재 경로를 포함해 최대 255자
- 파일 이름에는 영문자, 숫자, 붙임표(-), 밑줄(_), 마침표(.)를 주로 사용.
  - 윈도우: 특수문장 중 스페이스바와 &, ~ 등은 사용 가능, ₩, /, :, *, ?, ” ”, <, >, | 등은 사용 불가
  - 유닉스: 스페이스바 포함, 대부분의 특수문자 사용 불가. 대문자와 소문자 구분

1-2 파일

■ 파일 연결 프로그램

- 데이터 파일을 더블클릭하면 해당 파일을 사용하는 응용 프로그램이 실행되는데 이것을 연결 프로그램이라고 함
- 윈도우에서는 데이터 파일에 연결된 응용 프로그램 변경 가능
- 파일 확장자 변경도 가능. 그러나 확장자 바꾼다고 내용이 바뀌지는 않음

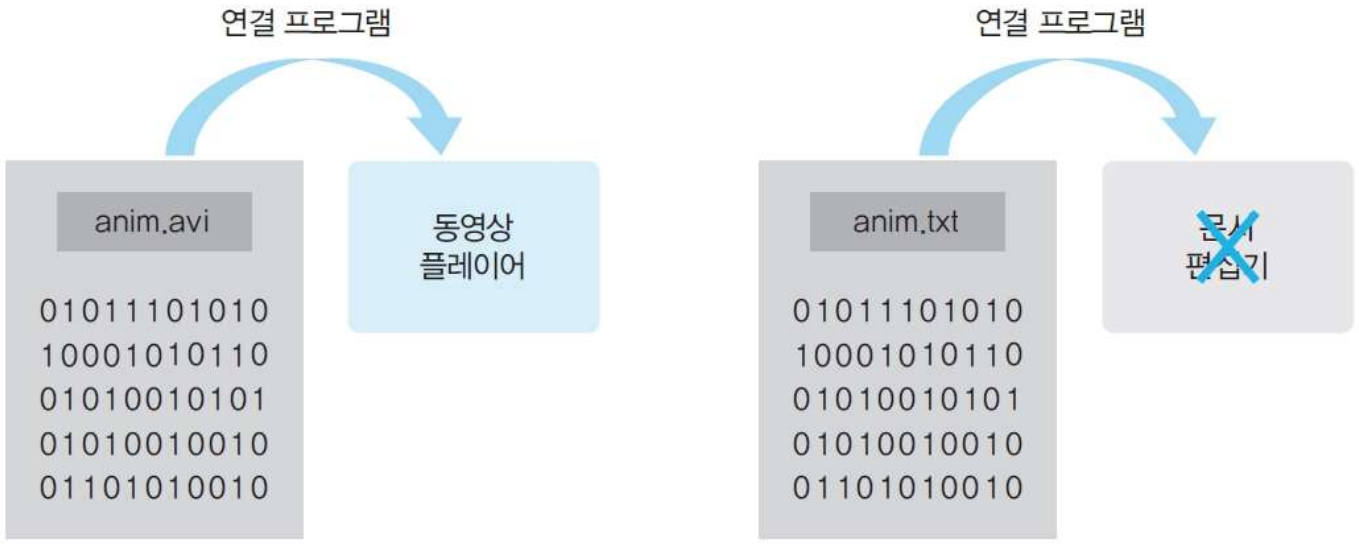

그림 11-4 데이터 파일과 연결 프로그램

1-2 파일

■ 파일 속성

표 11-3 파일 속성의 종류

속성	특징
name	파일 이름
type	파일 종류
size	파일 크기
time	파일에 대한 접근 시간
location	파일 위치
accessibility	파일에 대한 접근 권한
owner	파일 소유자

1-2 파일

■ 파일 작업

표 11-4 파일 자체를 변경하는 작업

작업	설명	작업	설명
open	파일을 연다.	copy	파일을 복사한다.
close	파일을 닫는다.	rename	파일 이름을 변경한다.
create	새로운 파일을 생성한다.	list	파일을 나열한다.
remove	파일을 이동한다.	search	파일을 찾는다.

표 11-5 파일 내용을 변경하는 작업

작업	설명	작업	설명
open( )	파일을 연다.	write( )	파일에 새로운 내용을 쓴다.
create( )	새로운 파일을 생성한다.	update( )	파일 내용 중 일부를 변경한다.
close( )	파일을 닫는다.	insert( )	파일에 새로운 내용을 추가한다.
read( )	파일 내용을 읽는다.	delete( )	파일 내용 중 일부를 지운다.

1-2 파일

■ 파일 헤더

- 파일 헤더(header)에는 파일의 버전 번호, 크기, 특수 정보 등 응용 프로그램이 필요한 정보 담김
- 모든 파일에 공통으로 적용되는 정보인 파일 속성은 파일 테이블에 위치, 해당 파일에 필요한 정보를 가지고 있는 파일 헤더는 파일의 맨 앞에 위치

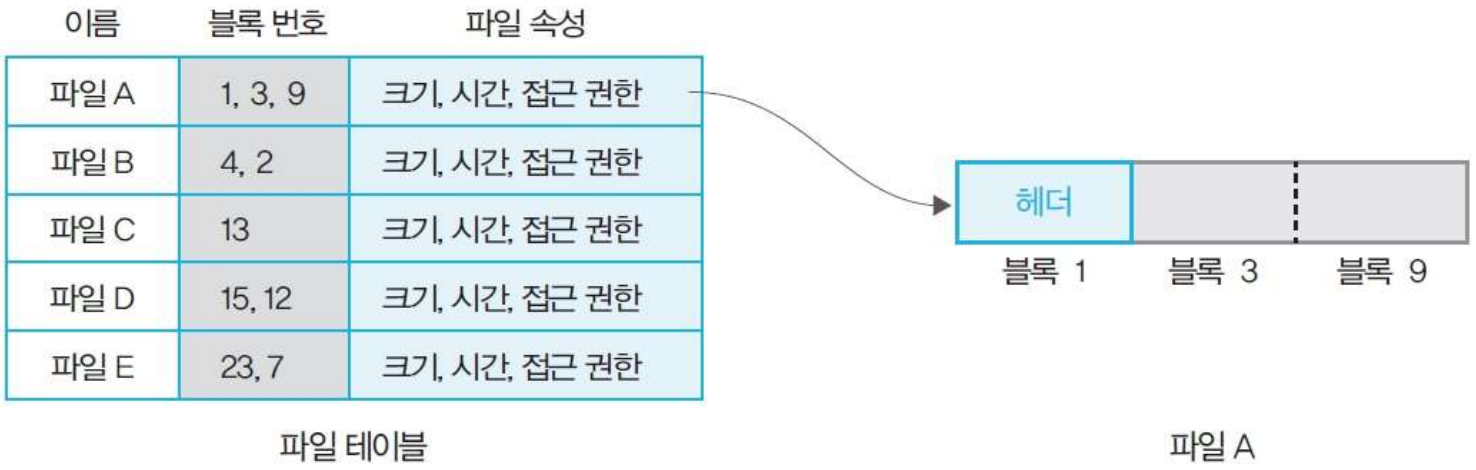

그림 11-5 파일 속성과 파일 헤더

1-3 저장장치 관리 기법

■ 파티션

- 저장장치를 2개 이상의 묶음으로 나누는 것으로 디스크를 논리적으로 분할
- 대용량 저장장치를 하나로 사용하기보다 여러 개로 나누면 관리하기 편리

그림 11-6 디스크와 파티션의 관계

1-3 저장장치 관리 기법

■ 포맷

- 포맷(format)은 저장장치의 파일 시스템을 초기화하는 작업
- 파일 테이블이 없는 저장장치를 포맷하면 파일 테이블이 새로 탑재, 있는 저장장치를 포맷하면 파일 테이블 초기화
- 빠른 포맷: 데이터는 그대로 둔 채 파일 테이블만 초기화
- 느린 포맷: 파일 테이블을 초기화할 뿐 아니라 블록의 모든 데이터를 0으로 만들어 포맷 시간이 많이 걸림.
- 저장장치를 포맷할 때 블록의 크기 지정 가능

1-3 저장장치 관리 기법

■ 조각 모음

- 파일 저장과 지우기를 반복하면 중간중간 빈 공간이 생기는데 이를 조각화 또는 단편화
- 하드디스크에 조각이 많이 생기면 큰 파일이 여러 조각으로 나뉘어 저장, 이를 읽으려면 하드디스크의 여러 곳을 돌아다녀야 하므로 성능 저하. 이를 방지하려면 주기적으로 조각 모음 필요
- USB 메모리나 SSD처럼 반도체를 이용한 저장장치는 조각 모음을 하지 않아도 성능의 차이가 없음

1-4 파일 구조

■ 순차 파일 구조(sequential file structure)

- 파일 내용이 하나의 긴 줄로 늘어선 형태
- 장점
  - 모든 데이터가 순서대로 기록되기 때문에 저장 공간에 낭비되는 부분이 없음
  - 구조가 단순하여 테이프는 물론 플로피디스크나 메모리를 이용한 저장장치에도 적용 가능
  - 순서대로 데이터를 읽거나 저장할 때 매우 빠르게 처리됨
- 단점
  - 파일에 새로운 데이터를 삽입하거나 삭제할 때 시간이 많이 걸림
  - 특정 데이터로 이동할 때 직접 접근이 어려워 앞에서부터 순서대로 움직여야 하므로 데이터 검색에 적당하지 않음

그림 11-11 순차 파일 구조

1-4 파일 구조

■ 인덱스 파일 구조(index file structure)

- 순차 파일 구조에 인덱스 테이블을 추가하여 순차 접근과 직접 접근이 가능
- 현대의 파일 시스템은 인덱스 파일 구조로, 순차 파일 구조로 파일 저장, 인덱스 테이블을 보고 원하는 파일에 직접 접근

그림 11-12 인덱스 파일 구조

1-4 파일 구조

■ 직접 파일 구조(direct file structure)

- 저장하려는 데이터의 특정 값에 어떤 관계를 정의하여 물리적인 주소로 바로 변환하는 파일 구조
- 특정 함수를 이용하여 직접 접근이 가능한 파일 구조로, 이때 사용하는 것이 해시 함수(hash function)

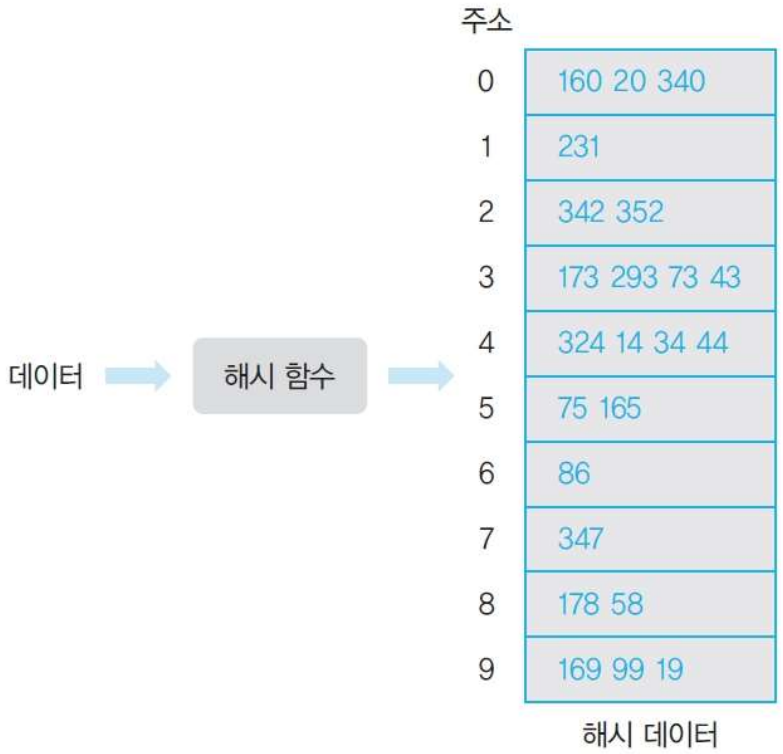

그림 11-13 직접 파일 구조

2-1 디렉터리의 개념

■ 디렉터리

- 관련 있는 파일을 하나로 모아놓은 곳
- 1개 이상의 자식 디렉터리와 1개 이상의 파일을 가질 수 있음
- 하나의 디렉터리에는 여러 개의 파일과 자식 디렉터리가 존재

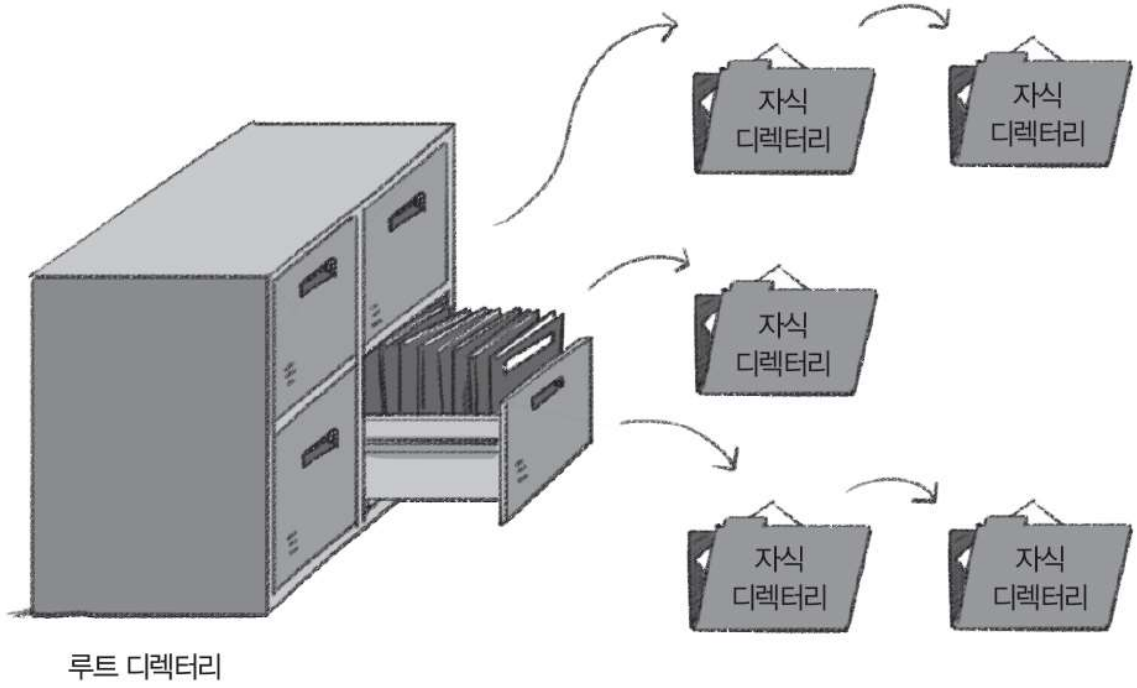

그림 11-14 디렉터리의 개념

2-1 디렉터리의 개념

■ 디렉터리의 계층 구조

- 디렉터리는 여러 층으로 구성
- 루트 디렉터리(root directory): 최상위에 있는 디렉터리

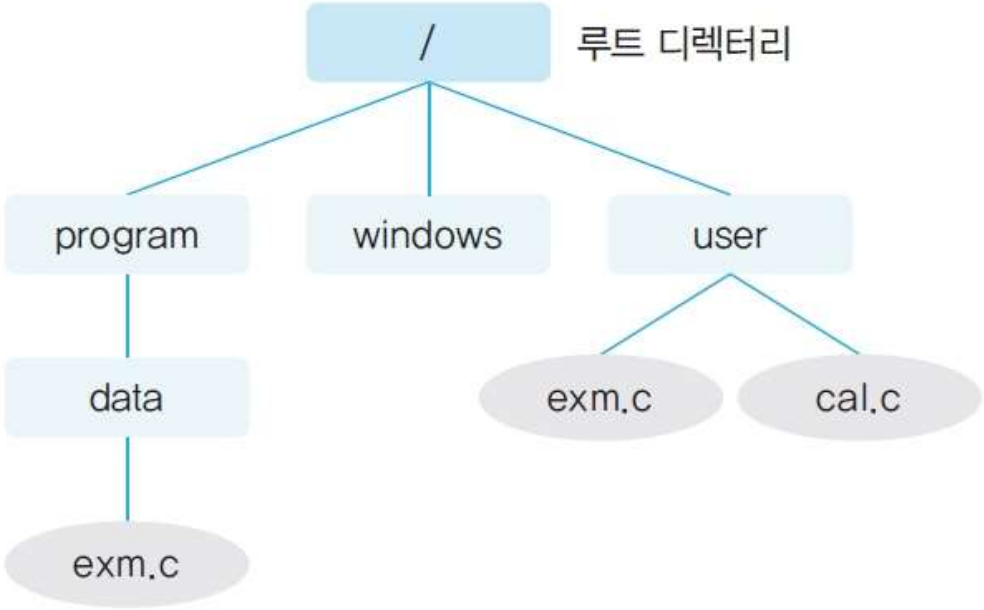

그림 11-15 디렉터리 계층 구조의 예

2-2 디렉터리 파일

■ 디렉터리 파일

- 디렉터리도 파일임
- 일반 파일에는 데이터가 담기고 디렉터리에는 파일 정보가 담김
- 디렉터리 헤더에는 디렉터리 이름, 만든 시간, 접근 권한 등의 정보가 기록됨

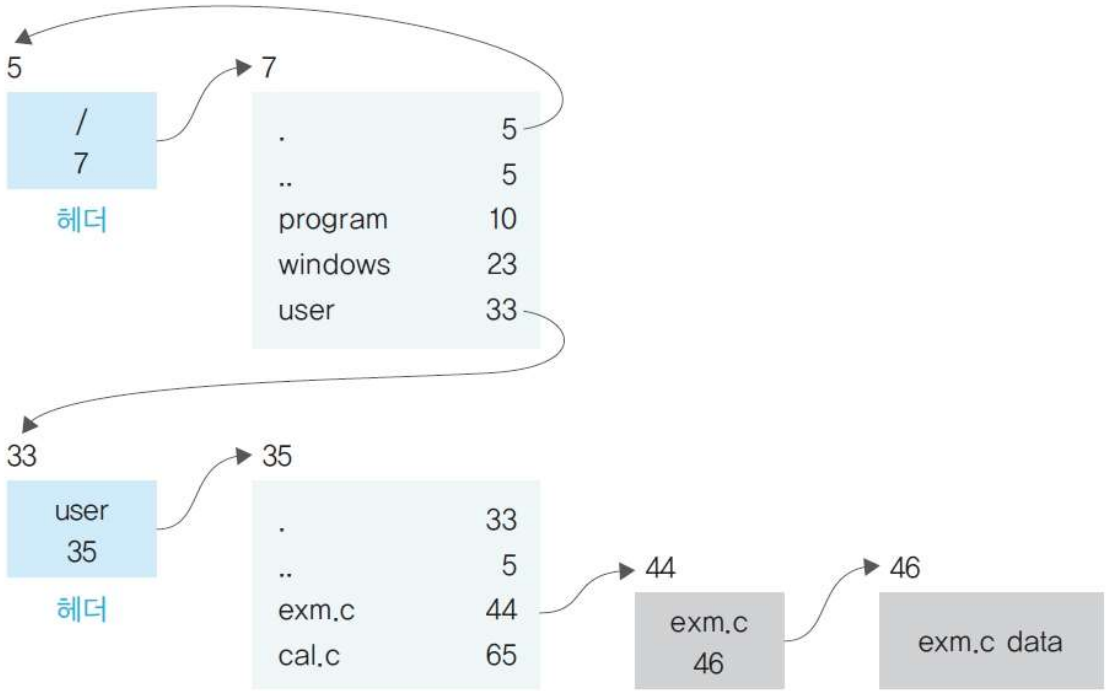

그림 11-16 루트 디렉터리와 user 디렉터리 파일 구조의 예

2-3 경로

■ 경로

- 파일이 전체 디렉터리 중 어디에 있는지를 나타내는 정보
- 한 디렉터리에는 같은 이름을 가진 파일이 존재할 수 없지만, 서로 다른 디렉터리에는 같은 이름의 파일이 존재할 수 있음
- 절대 경로(absolute path): 루트 디렉터리를 기준으로 파일 위치 표시
- 상대 경로(relative path): 현재 위치를 기준으로 파일 위치 표시

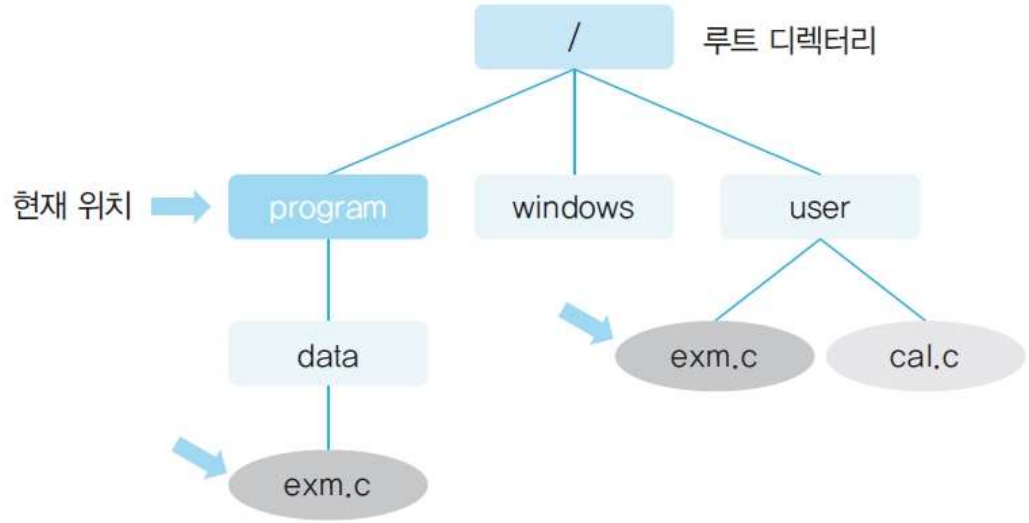

그림 11-17 경로의 개념

2-4 디렉터리 구조

■ 1단계 디렉터리 구조

- 루트 디렉터리에 새로운 디렉터리를 만들 수 있지만 디렉터리 안에 자식 디렉터리를 만들 수 없음
- 최대 1단계만 구현할 수 있으며 디렉터리 안에는 파일만 존재
- 매우 단순하지만 파일 많아지면 사용하기 불편

그림 11-18 1단계 디렉터리 구조의 예

2-4 디렉터리 구조

■ 다단계 디렉터리 구조

- 루트 디렉터리를 시작점으로 여러 단계의 디렉터리가 가지처럼 뻗음. 트리 디렉터리 구조라고도 함
- 단계 확장에 제약이 없고 디렉터리에 파일과 디렉터리를 둘 다 저장할 수 있음

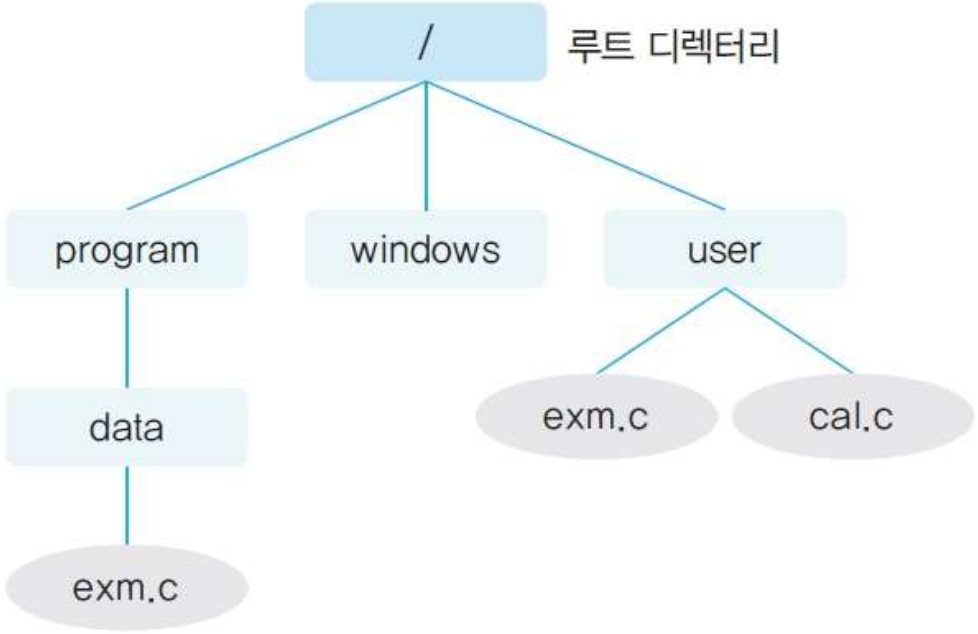

그림 11-19 다단계 디렉터리 구조의 예

2-4 디렉터리 구조

■ 바로가기 링크를 포함한 디렉터리 구조

- 기본적으로는 트리(tree) 구조이나 '바로가기 링크'로 인하여 현재의 디렉터리 구조는 그래프 구조로 바뀜

* 사이클이 없는 그래프를 트리(tree)라 함.

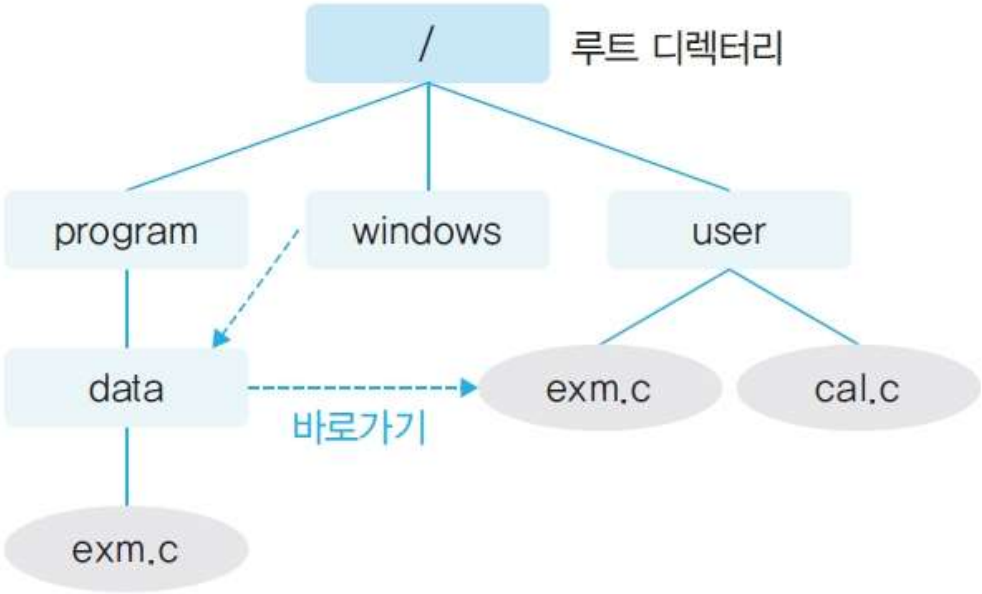

그림 11-21 그래프 디렉터리 구조

2-5 마운트

■ 마운트

- 유닉스에서 여러 개의 파티션을 통합하는 명령어

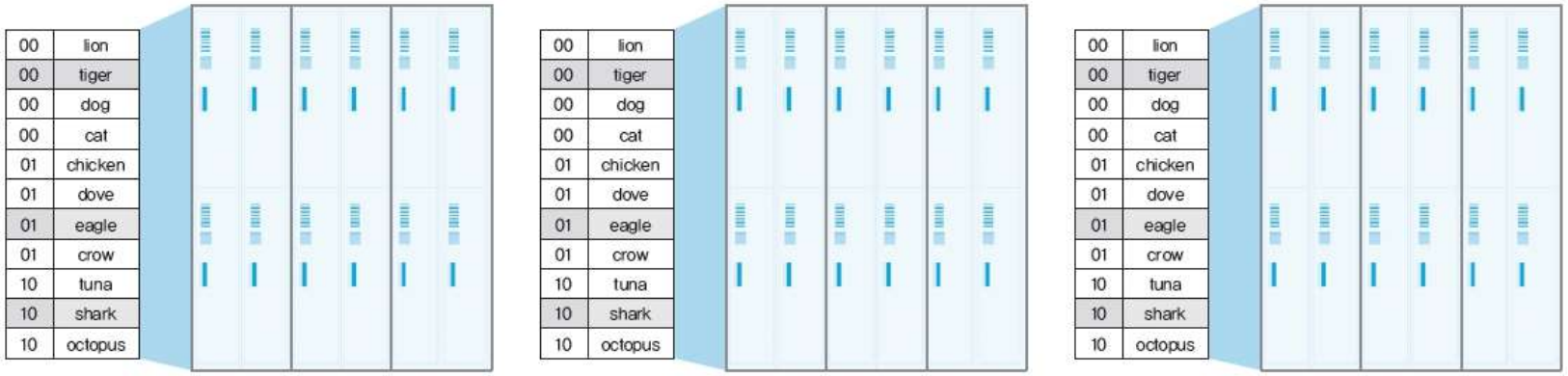

그림 11-22 독립 파티션

2-5 마운트

■ 마운트

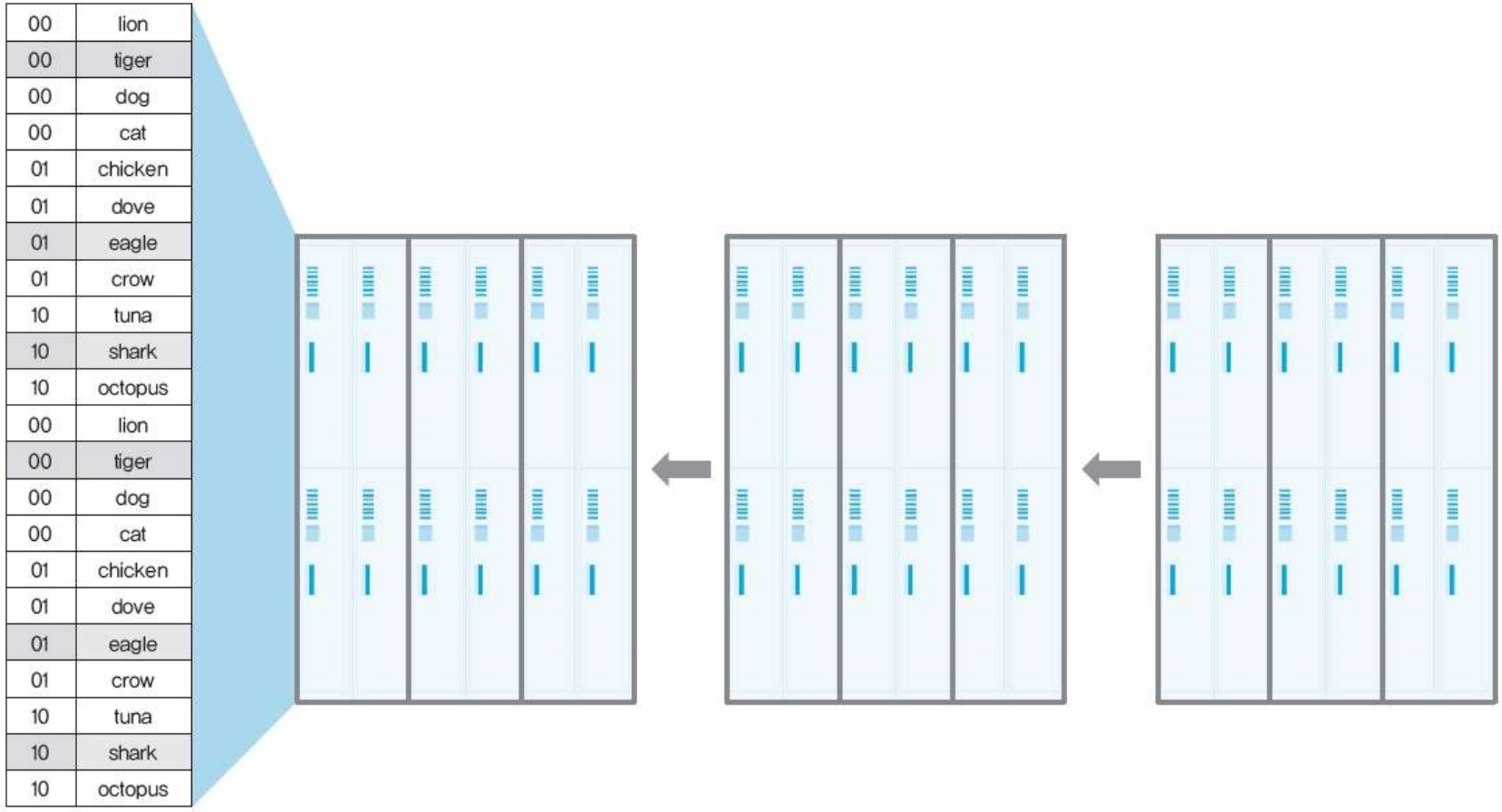

그림 11-23 마운트의 개념

2-5 마운트

■ 마운트를 이용해 2개 파티션을 하나의 파일 시스템으로 만든 예

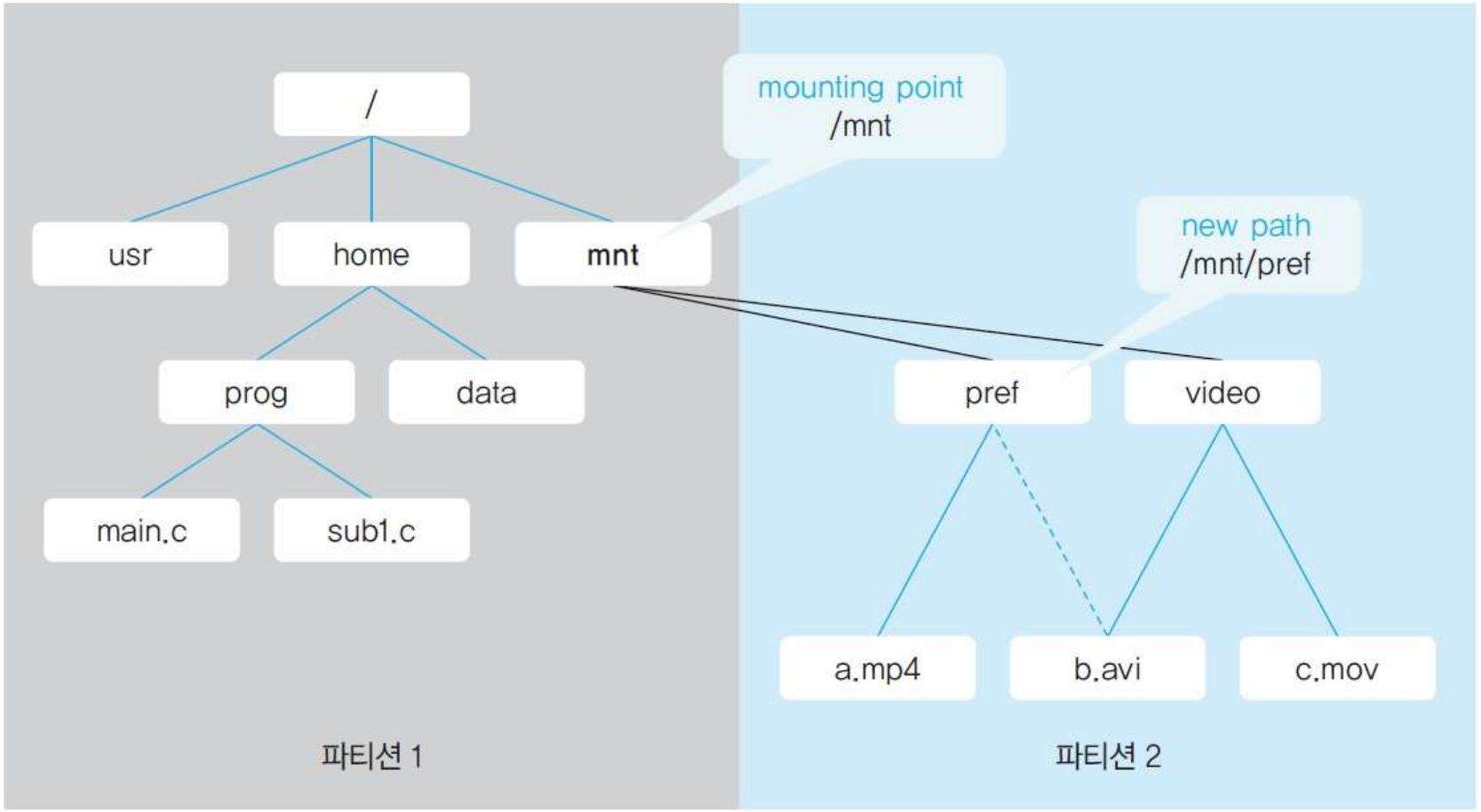

그림 11-24 유닉스의 마운트

3-1 할당 방식

■ 연속 할당과 불연속 할당

- 연속 할당(contiguous allocation) 방식
  - 파일을 구성하는 데이터를 디스크상에 연속 배열하는 간단한 방식
  - 파일을 저장하거나 삭제하면 빈 공간이 생기는데, 디스크에 남은 공간 중 파일 크기와 맞는 연속 공간이 없으면 연속 할당이 불가능하므로 실제로는 사용되지 않음

- 불연속 할당(non-contiguous allocation) 방식
  - 비어 있는 블록에 데이터를 분산 저장하고 이에 관한 정보를 파일 시스템이 관리하는 방식
  - 연결 리스트를 이용한 연결 할당과 인덱스를 이용한 인덱스 할당이 대표적

3-1 할당 방식

■ 연결 할당(linked allocation)

- 파일에 속한 데이터를 연결 리스트로 관리하는 방식
- 파일 테이블에는 시작 블록에 대한 정보만 저장, 나머지 데이터는 시작 블록부터 연결하여 저장
- 체인으로 연결한 것처럼 보여서 체인 할당(chained allocation)이라고도 함
- 연결 할당 방식의 예: 윈도우의 FAT(File Allocation Table)

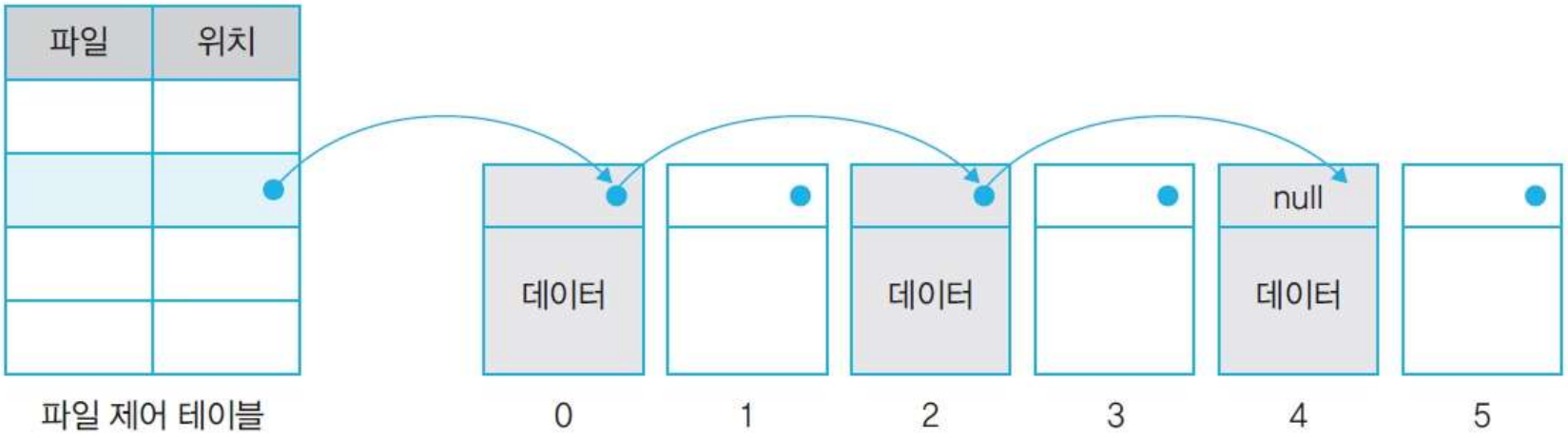

그림 11-25 연결 할당 방식

3-1 할당 방식

■ 연결 할당 방식의 예(FAT)

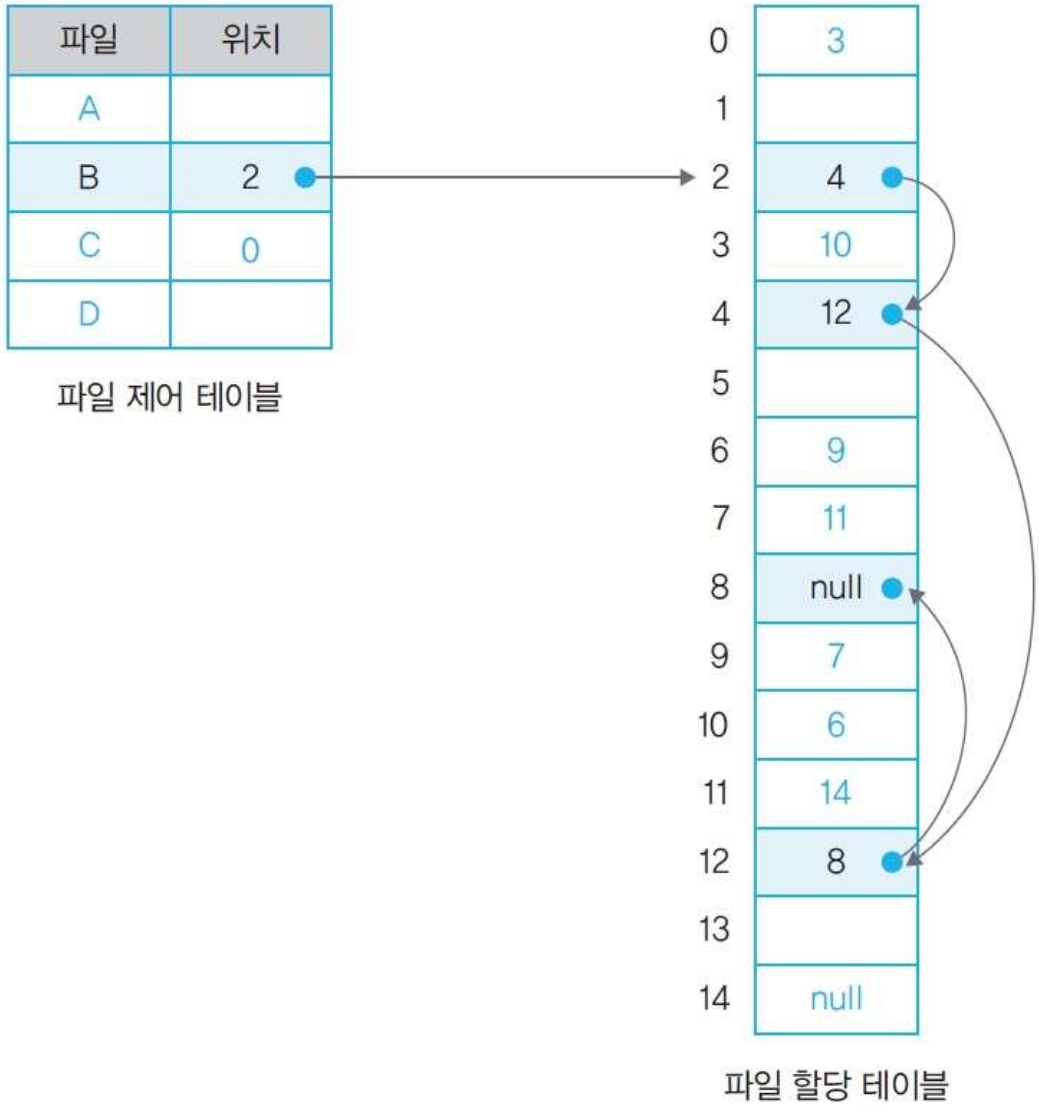

그림 11-26 파일 테이블을 이용한 불연속 할당

3-1 할당 방식

■ 인덱스 할당(indexed allocation)

- 테이블의 블록 포인터가 데이터 블록을 연결하는 것이 아니라, 데이터의 인덱스를 담고 있는 인덱스 블록을 연결
- 인덱스 블록은 실제 데이터의 위치에 관한 정보를 순서대로 보관
- 테이블이 꽉 차서 더 이상 데이터를 연결할 수 없을 때는 인덱스 블록을 연결하는 간접 인덱스 블록(indirect index block)으로 테이블을 무한 확장

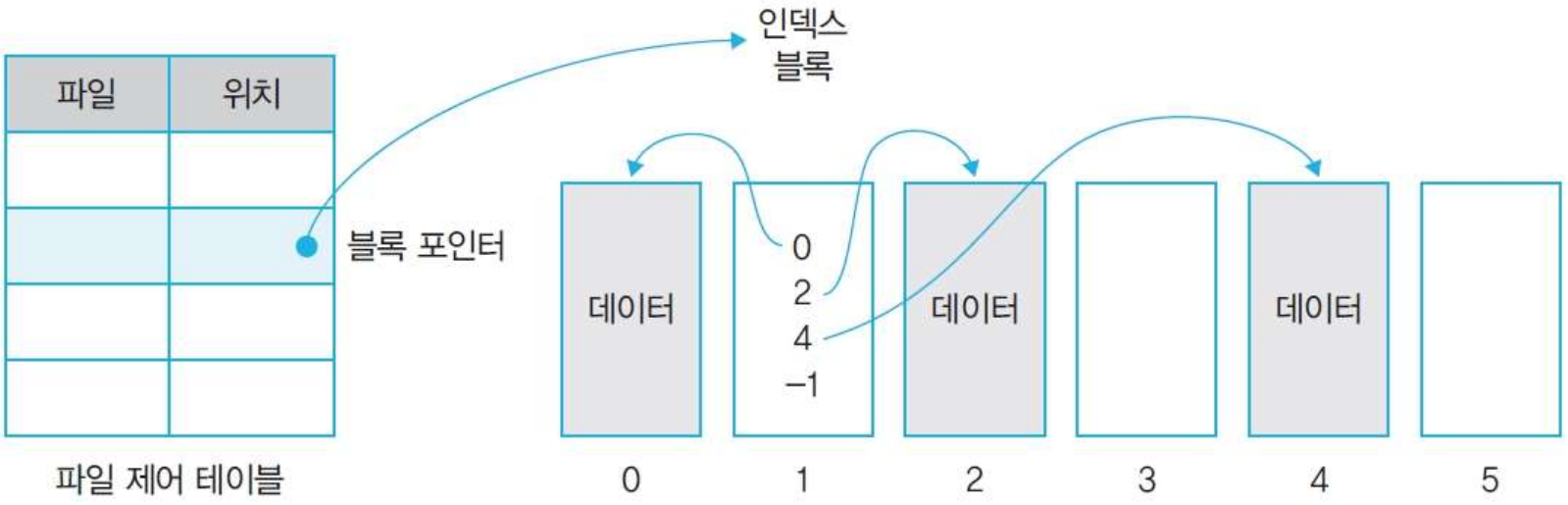

그림 11-27 인덱스를 이용한 불연속 할당

3-1 할당 방식

■ 인덱스 할당 방식의 예(아이노드)

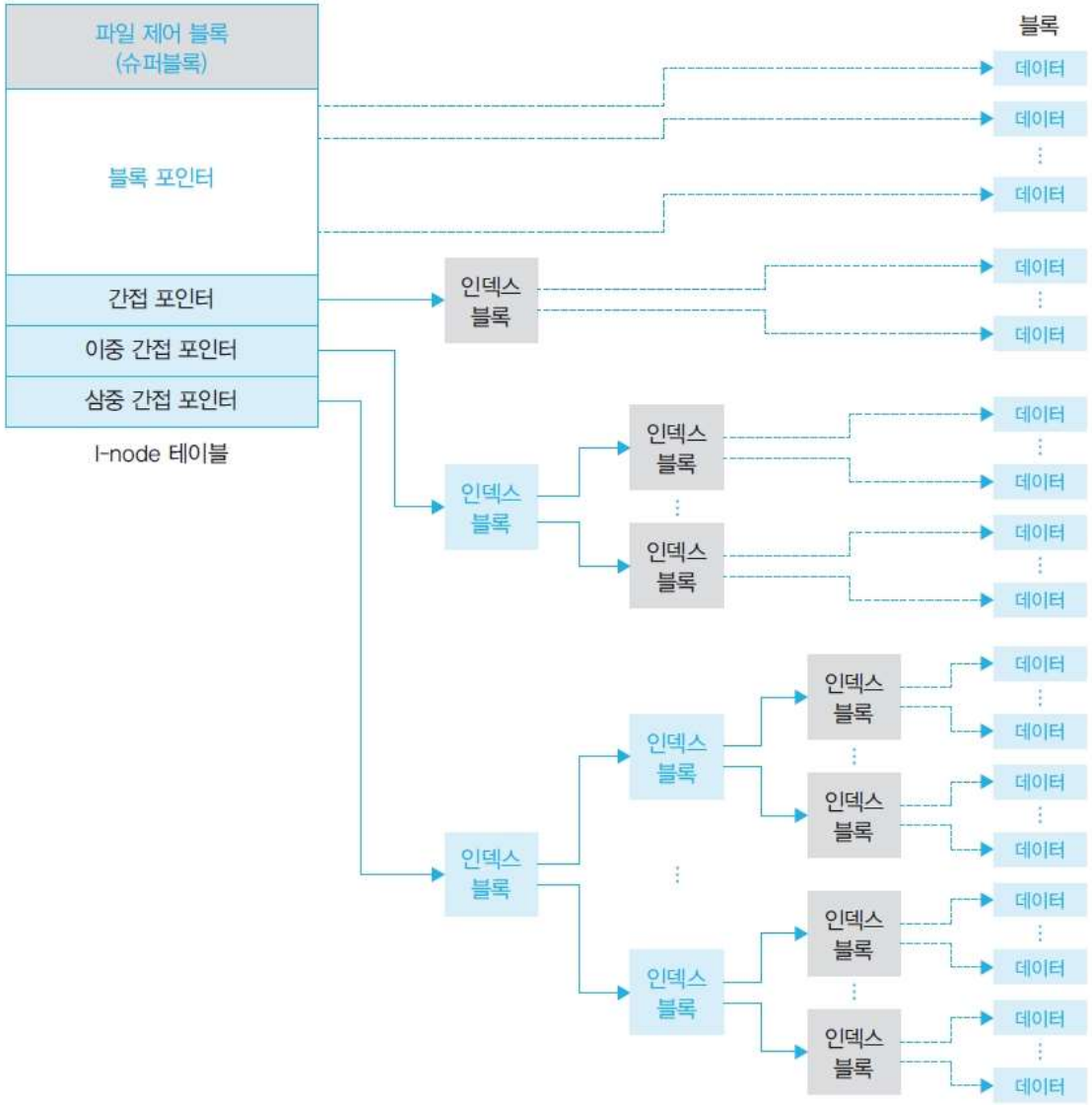

그림 11-28 아이노드 파일 시스템의 구조

3-2 빈 공간 관리

■ 빈 공간 리스트(free block list)

- 파일 시스템은 디스크의 내부 단편화를 줄이고 빈 공간을 효율적으로 관리하기 위해 빈 블록의 정보만 모아놓은 빈 공간 리스트를 유지
- 파일 시스템에서는 파일 테이블의 헤더를 삭제하고 사용했던 블록을 빈 공간 리스트에 등록하는 것을 파일 삭제로 간주
- 어떤 데이터를 지우고 새로운 데이터를 디스크에 넣을 때 방금 지워진 블록에 할당하는 것이 아니라 리스트에 있던 블록 중 맨 앞에 있는 블록에 할당

3-2 빈 공간 관리

■ 빈 공간 리스트(free block list)

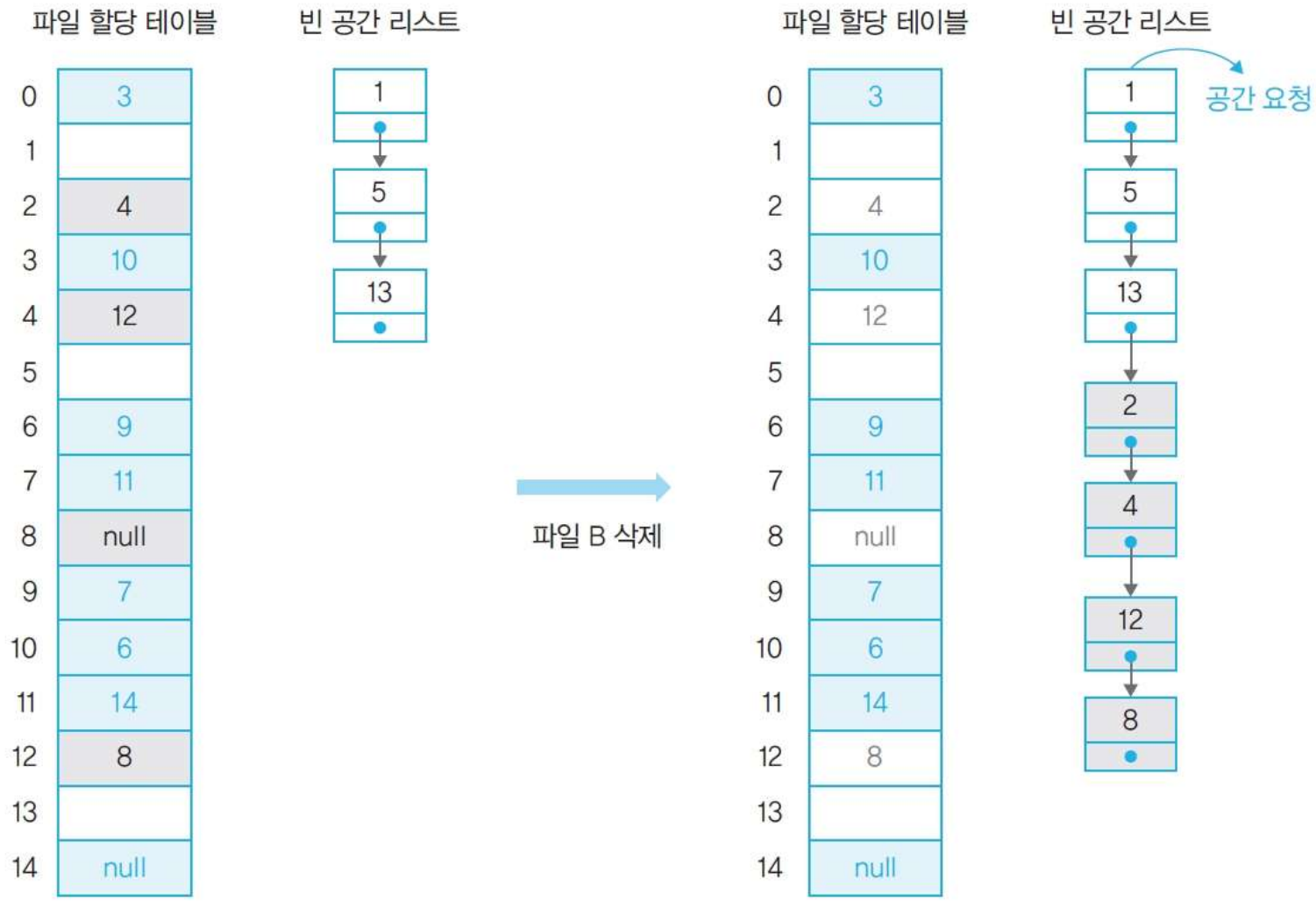

그림 11-29 빈 공간 리스트의 변화

4-1 유닉스의 실행 파일

■ 유닉스 파일 시스템의 접근 패턴

- 접근 패턴의 맨 앞자리는 파일의 종류
- 나머지 아홉 자리는 rwx라는 세 덩어리로 구성
- rwx 덩어리 중
  - 첫 번째 덩어리 : 파일의 소유자(owner) 권한 부여
  - 두 번째 덩어리 : 소유자가 속한 그룹(group) 권한 부여
  - 세 번째 덩어리 : 소유자도 아니고 같은 그룹도 아닌 제삼자(others) 권한 부여

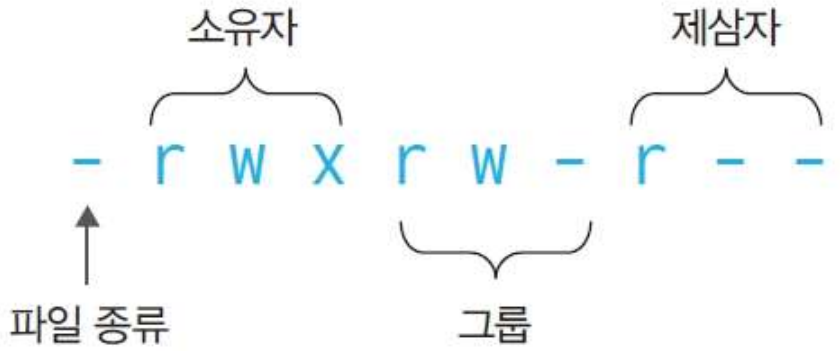

그림 11-30 유닉스의 파일 접근 패턴

4-1 유닉스의 실행 파일

■ chmod

- 유닉스에서 접근 패턴을 변경할 때 사용하는 명령어
- 접근 패턴에 숫자를 부여하여 변경
  - read는 4, write는 2, execute는 1
  - chmod 명령을 이용하여 살리고 싶은 숫자를 모두 더하면 됨

(a) 모든 접근 패턴 허용

(b) 일부 접근 패턴만 허용

그림 11-31 유닉스의 접근 패턴 숫자