10장. 파일 시스템

목표

- 파일 시스템의 기능 설명
- 파일 시스템 인터페이스의 특징 기술
- 파일 시스템 설계 절충안(tradeoff) 논의
  - 파일 접근 방법, 파일 공유, 파일 잠금, 디렉토리 구조 등
- 파일 시스템 보호 방법 소개

10.1 파일 개념

- 정보 저장장치
  - 자기 디스크, 자기 테이프, 광 디스크, 플래시 메모리 …
- 파일(file)
  - 운영체제가 정보 저장장치의 물리적 특성을 추상화한 논리적 저장 단위
    - 정보 저장장치에 대한 일관된 논리적 관점(unique logical view)을 제공
  - 보조저장장치에 저장된 관련된 정보의 집합 – 이름이 부여됨
    - 운영체제가 물리적 저장장치로 맵핑하여 정보 접근
- 파일 구조
  - 없음 – 바이트 또는 워드의 연속
  - 단순 레코드 구조 – 줄, 고정길이, 가변 길이 레코드의 연속
  - 복잡한 구조 – formatted 문서, relocatable load file
- 운영체제에서는 파일 구조는 byte의 연속으로 다룸 – 대단히 일반적
  - 파일 구조의 의미는 생성한 응용 프로그램과 사용자에 의해서 정의됨
    - 이진 실행 파일 : 운영체제
    - 대부분의 파일 : 관련된 응용 프로그램

파일 속성 및 연산

- 파일 속성(attributes)
  - 이름, 식별자(identidier), 유형, 위치, 크기, 보호, 날짜시간, 사용자 등
- 디렉토리 구조
  - 보조저장장치에 저장되어 모든 파일에 대한 정보를 유지함
  - 디렉토리 항목 – 파일 이름, 다른 파일 속성을 찾기 위한 식별자로 구성
- 파일 연산
  - **기본 연산** - 생성(create), 쓰기(write), 읽기(read), 위치 재설정 (reposition, seek), 삭제(delete), 절단(truncate – 길이 0인 파일로 만듬)
  - **기타 연산** - 추가(append), 복사, 이름변경, 속성 획득/설정 등 …
  - **open 연산** - 메모리에 있는 open file table에 파일에 대한 정보를 등록하고 인덱스를 반환
    - 파일이름(인수) → 디렉토리 검색 → 파일에 대한 디렉토리 항목을 open file table로 복사 → 요청한 접근 모드(인수)과 파일의 접근 모드 비교 → 허용되면 open file table에 대한 index를 반환
  - **close 연산** – open file table에 있는 항목을 제거

open file 테이블 및 open 파일 잠금

- 2단계 open file 테이블 - 다수의 프로세스가 동시 open 지원
  - **프로세스 단위 테이블**(per-process table)
    - 프로세스 종속 정보 - 현재 file position pointer, access right 등
    - system-wide table에 대한 포인터
  - **전체 시스템 테이블**(system-wide table)
    - 프로세스 독립 정보 - file open count, disk location, file size 등
    - file open count가 0이 되면 파일테이블에서 제거됨
- 오픈 파일 잠금(lock)
  - 여러 프로세스가 공유하는 파일에 대해 사용
    - **shared lock** – reader lock과 유사 (여러 프로세스가 동시 획득가능)
    - **exclusive lock** – writer lock과 유사 (한번에 한 프로세스만 획득가능)
  - 강제적 잠금(mandatory lock)과 권고적 잠금(advisory lock)
    - **강제적 잠금**: 운영체제가 locked file에 대한 접근을 막음
      - windows에서 사용, deadlock이 발생하지 않도록 동기화에 유의
    - **권고적 잠금**: 운영체제는 잠금상태를 알려주고, 잠금은 개발자 몫임
      - unix에서 사용

파일 유형

- 파일 유형 구분
  - DOS, Windows - 확장자
  - Mac OS – 생성한 응용프로그램 속성
  - UNIX – magic number(파일 앞부분 내용),
    확장자를 OS가 강요하지 않음. 확장자 사용은 응용 프로그램 몫

file type	usual extension	function
executable	exe, com, bin or none	read to run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter

확장자

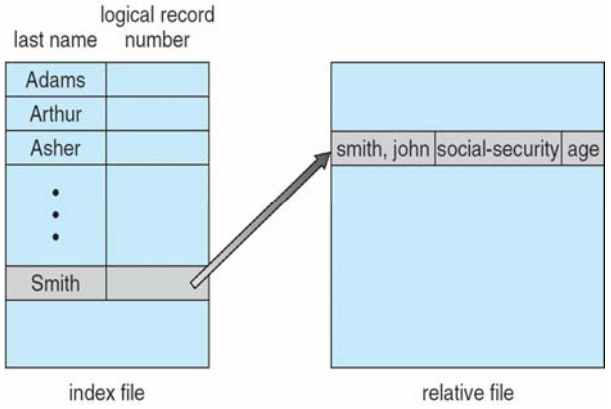
ELF executable:	**.ELF**
UNIX script :	#!
PDF:	%PDF
Postscript(PS):	%!
JPG:	ff d8 ff e0 (hex)

magic number

10.2 접근 방법(Access Methods)

- 순차 접근(sequential access) – tape 모델
  - 순서대로 접근
  - 연산: read next, write next, position to n
- 직접 접근 (direct access) – disk 모델
  - 임의 순서로 접근 허용(random access)
  - 연산: read n, write n
  - 블록 번호 n은 파일의 시작을 0으로 보고 계산한 상대블록번호임
    → 직접접근 방법을 상대접근(relative access)이라고도 함
- 인덱스 순차 접근 (indexed sequential access) - ISAM
  - (key, data)형식의 레코드들을 순서대로 정렬
  - 파일에 대한 index 파일 생성 → 메모리에 유지할 수 있음
  - 1차로 index 검색, 2차로 pointer를 사용하여 파일을 직접 접근
  - index가 메모리에 저장할 수 없을 정도로 파일이 커지면
    → 2차 index 파일 (index 파일의 index) 생성

인덱스 순차 접근 - 색인파일과 상대파일



10.3 디렉터리와 디스크 구조

- 시스템은 매우 많은 수의 파일을 디스크에 저장함
  - 저장된 파일을 관리하기 위해서 체계적인 구성(organization)이 필요
- 저장장치와 파일시스템
  - **디스크**를 여러 부분으로 분할 → **파티션**(partition)
  - 각 파티션에 **file system**을 생성
  - file system은 **directory 구조**를 사용하여 directory들을 생성
  - **directory**는 디렉토리에 있는 파일에 대한 정보를 저장
- 디렉터리
  - **파일 이름**을 **파일 위치**로 변환하는 정보를 갖는 심볼 테이블
  - 각 디렉터리 항목은 파일 속성을 저장함

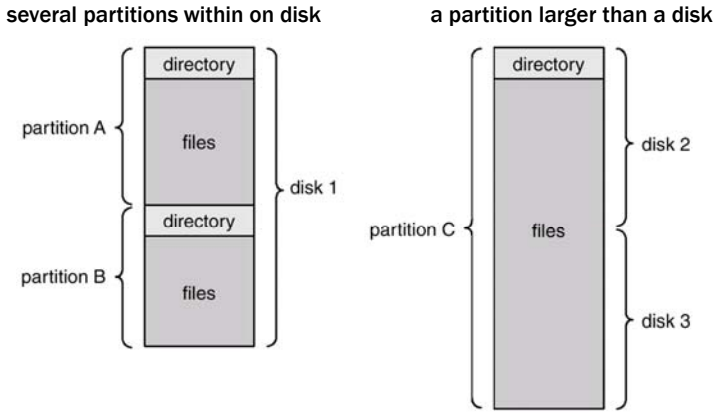
디스크 구조

- 디스크 구조
  - disk는 여러 개의 partition으로 분할 가능
  - disk / partition은 다음 두 형태로 사용
    - raw (unformatted) – without a file system (예) swap 공간 등
    - formatted with a file system
- 볼륨(volume)
  - file system을 포함한 개체(entity)
  - 전체 장치, 장치의 부분집합 또는 RAID로 연결된 다수 장치일 수 있음
  - 각 volume은 논리적 가상 디스크(virtual disk)로 취급될 수 있음
- special purpose 파일 시스템 – 보통 파일이 아닌 특수 용도로 사용

  Solaris에서
  - objfs – 커널 심볼 값 제공하는 가상 파일 시스템
  - procfs – 프로세스에 대한 정보를 제공하는 가상 파일 시스템
  - tmpfs – 메모리에 생성되는 임시 파일 시스템
  - lofs – 다른 파일 시스템에 접근할 수 있게 하는 loopback 파일 시스템

전형적인 파일 시스템 구성

- 각 partition에 디렉토리 구조와 파일이 저장됨

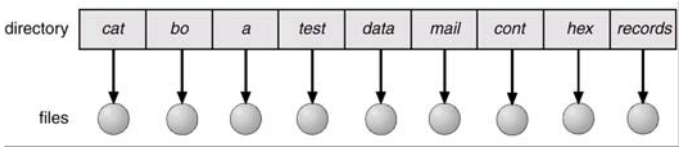


디렉터리의 논리적 구조

- 디렉터리 구조
  - 파일 시스템은 디렉터리들의 조직하기 위해 디렉터리 구조 사용
- 디렉터리 구조 구성 시 고려 사항
  - 효율(efficiency) – 파일 위치를 신속하게 검색
  - 명명(naming) – 사용자가 편리하게
    - 여러 사용자가 다른 파일에 대해 같은 이름을 사용 가능
    - 같은 파일이 여러 개의 다른 이름 사용 가능
  - 그룹핑(grouping) – 파일 특성에 따른 논리적 그룹핑
- 디렉터리의 논리적 구조
  - 1단계 디렉터리(single-level directory)
  - 2단계 디렉터리(two-level directory)
  - 트리 구조 디렉터리(tree-structured directory)
  - 비순환 그래프 디렉터리(acyclic-graph directory)

Single-Level Directory

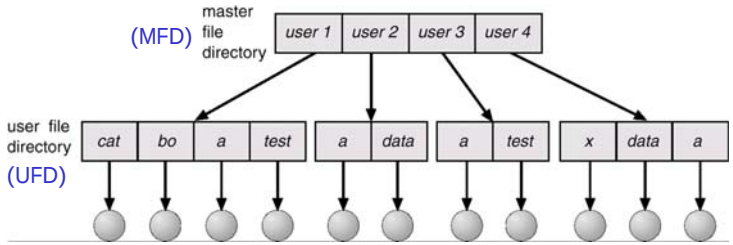
- 모든 사용자가 한 개의 디렉터리 사용



- 단점
  - naming problem
  - grouping problem

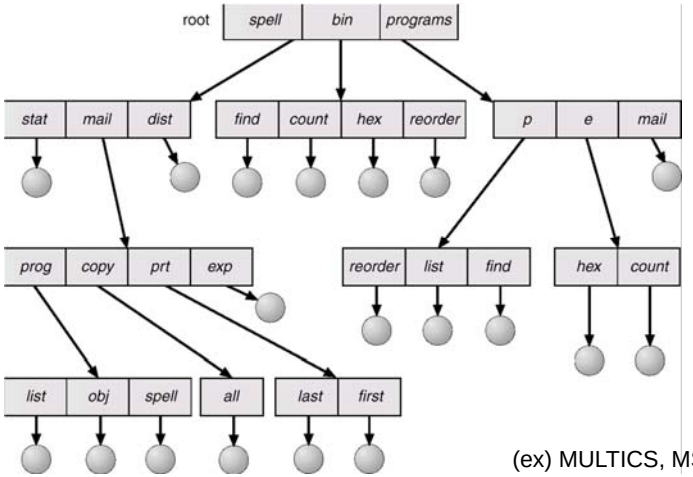
Two-Level Directory

- 각 사용자가 분리된 디렉토리를 사용



- 특징
  - 경로이름 :
    - /사용자이름/파일이름 (ex) /user1/test
  - 다른 사용자가 같은 이름 사용 가능
  - 효율적 파일 검색
  - no grouping capability

Tree-Structured Directories



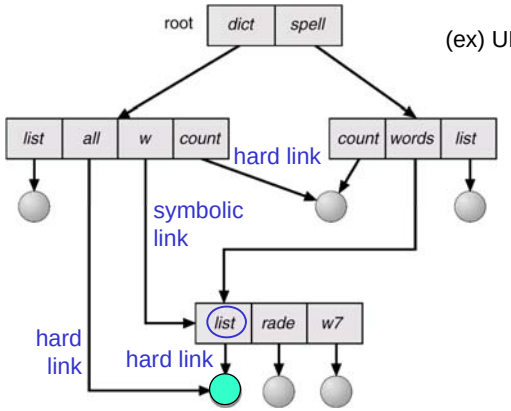
(ex) MULTICS, MS-DOS

Tree-Structured Directories (계속)

- 특징
  - 효율적 파일 검색
  - Grouping capability
  - 경로이름(Path name)
    - absolute path name (ex) /spell/mail/prt/first
    - relative path name (ex) prt/first, ../dist
- 현재 디렉터리(작업 디렉터리)
  - 프로세스는 현재 (작업) 디렉터리를 가짐
  - 파일을 참조할 때에 파일이름만 사용하면 현재 디렉터리를 검색
  - 작업 디렉터리를 변경할 수 있음 – cd 명령어
  - 사용자가 로그인을 할 때에 초기 작업 디렉터리(홈 디렉터리)에 위치

Acyclic-Graph Directories

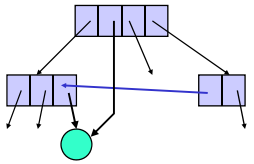
- 트리 구조의 일반화
- 디렉토리 간의 파일과 서브디렉토리 공유를 허용.
  (cf) Acyclic graph: a graph with no cycle



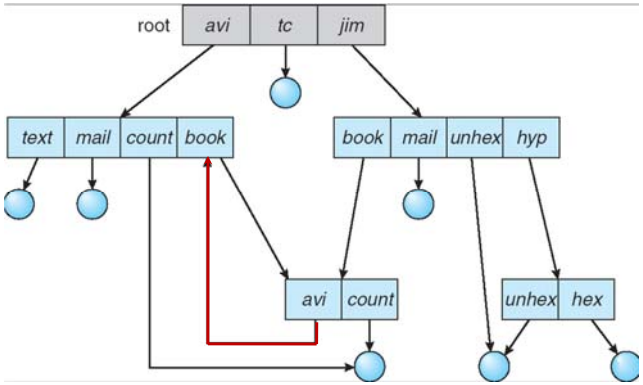
(ex) UNIX

Acyclic-Graph Directories (계속)

- Symbolic link와 hard link
  - symbolic link – 다른 파일/디렉터리에 대한 포인터 저장 ➔ indirect link
  - hard link – 저장 장치의 파일 위치를 직접 저장 ➔ direct link
- 같은 파일이 여러 개의 경로 이름 가질 수 있음 (aliasing)
  - 백업 등의 목적으로 전체 파일 시스템 순회 시에 중복 탐색 문제가 발생
  - (예-그림) /spell/word/list, /dict/all (hard link), /dict/w (symbolic link)
- 공유 파일의 삭제
  - hard link:
    - 직접 링크 삭제, reference count 유지
    - count=0이 되면 파일 삭제
  - symbolic link:
    - 심볼릭 링크만 삭제
    - 링크된 원본 file이 삭제되면
      링크는 존재하지 않는 파일을 가리킴

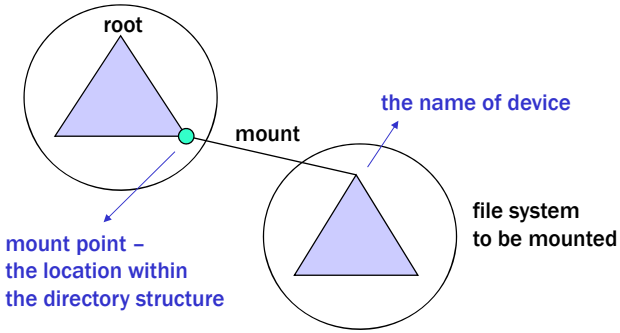


General Graph Directory



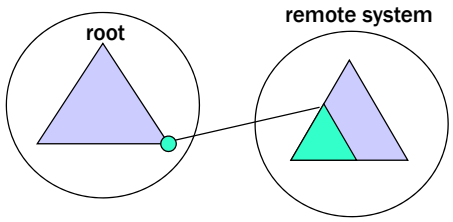
10.4 File-System Mounting

- 파일 시스템 mounting
  - 보조 기억장치에 설치된 파일 시스템을 접근할 수 있도록
    현재 파일시스템의 특정 디렉터리에 연결하는 것
- mount 절차
  - **디바이스 이름**과 **mount point**(빈 디렉터리)를 운영체제에게 제공
  - 운영체제는 디바이스가 유효한 파일시스템을 포함하고 있는지 검증함
  - 파일시스템은 지정된 mount point에 마운트 됨.



10.5 File 공유

- 다수의 사용자의 파일 공유
  - 다수의 사용자에 대한 파일 공유 기능이 바람직함.
  - 공유는 보호 방식을 통하여 이루어질 수 있음
    - owner (user), group
- 원격 파일 시스템(Remote File Systems)
  - 분산 시스템에서 파일은 네트워크를 경유하여 공유 가능
  - Network File System (NFS) 또는 Distributed File System(DFS)



File 공유

- **원격 파일 공유 구현 - Client-server** 모델을 사용하여 구현
  - Server는 여러 개의 client에게 서비스 제공가능
  - **NFS** – 표준 UNIX client-server 파일 공유 프로토콜
  - **CIFS** (Common Internet file system) – 표준 윈도우 파일 공유 프로토콜
  - 표준 운영체제의 파일에 대한 호출은 원격 파일에 대한 호출로 변환
- 분산 정보 시스템(Distributed Information Systems
  - 원격 컴퓨팅에 필요한 정보의 통합된 접근을 구현
  - (ex) LDAP (light-weight directory access protocol),
    DNS(domain name system),
    NIS (network information service),
    Domain, Active Directory – Windows

10.6 보호

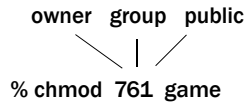
- 접근 제어
  - 파일 소유자/생성자의 접근 권한 제어
    - what can be done (무엇을 ?)
    - by whom (누가?)
- 접근 유형
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

접근 리스트(Access Lists)와 그룹 (UNIX)

- 접근 모드: read, write, execute
- 세 종류의 사용자 클래스

	r	w	x	
a) owner access	1	1	1	(7)
b) groups access	1	1	0	(6)
c) public access	0	0	1	(1)

- 파일이나 디렉터리에 대한 접근 권한 정의



(예) UNIX Directory Listing과 파일 허가권

-rw-rw-r--	1 pbg	staff	31200	Sep 3 08:30	intro.ps
drwx------	5 pbg	staff	512	Jul 8 09.33	private/
drwxrwxr-x	2 pbg	staff	512	Jul 8 09:35	doc/
drwxrwx---	2 pbg	student	512	Aug 3 14:13	student-proj/
-rw-r--r--	1 pbg	staff	9423	Feb 24 2003	program.c
-rwxr-xr-x	1 pbg	staff	20471	Feb 24 2003	program
drwx--x--x	4 pbg	faculty	512	Jul 31 10:31	lib/
drwx------	3 pbg	staff	1024	Aug 29 06:52	mail/
drwxrwxrwx	3 pbg	staff	512	Jul 8 09:35	test/

Windows 7 Access-Control List