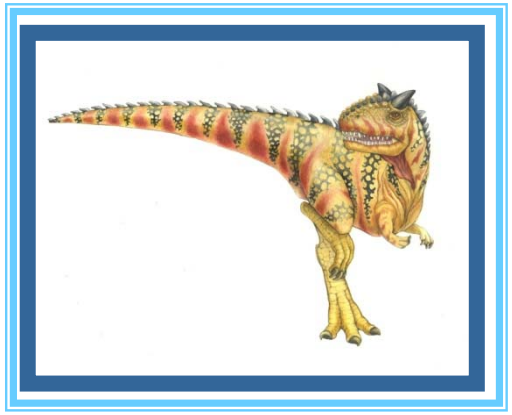
Chapter 11:
파일시스템 구현
File-System Implementation



Chapter 11: Implementing File Systems

- 파일시스템 구조 **File-System Structure**
- 파일시스템 구현 **File-System Implementation**
- 디렉터리 구현 **Directory Implementation**
- 할당 방법 **Allocation Methods**
- 자유공간 관리 **Free-Space Management**
- 효율성과 성능 **Efficiency and Performance**
- 회복 **Recovery**
- 로그 구조화된 파일시스템 **Log-Structured File Systems**
- 네트워크 파일시스템 **NFS**
- 예**: WA**리 파일시스템 **Example: WAFL File System**

11.1 File-System Structure

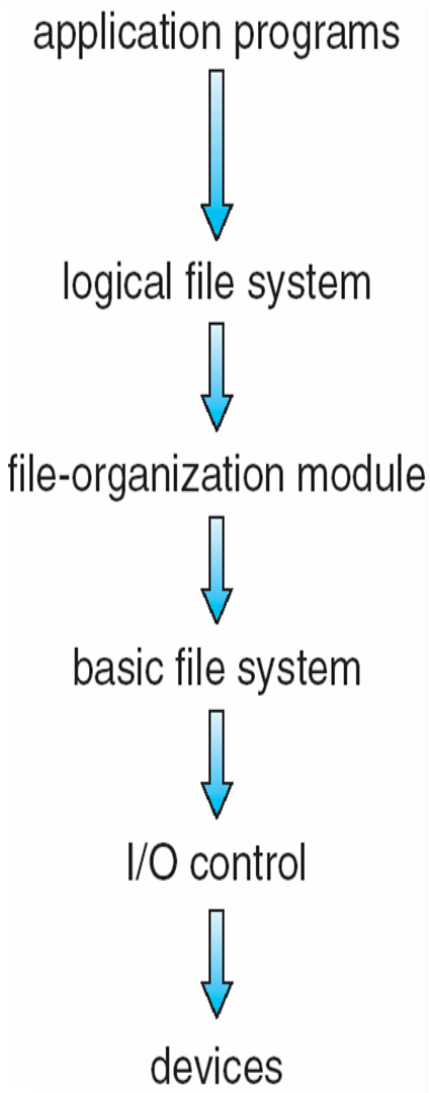
- 파일시스템은 쉽게 데이터를 저장하고, 찾고 또한 인출할 수 있게 함으로써 디스크를 보다 효율적이고 편리하게 사용할 수 있게 한다.
  - 파일 시스템이 사용자에게 어떻게 보여야 할지를 정의해야 한다.
    - 파일과 속성, 허용연산, 디렉터리구조의 정의를 포함
  - 논리파일을 물리저장장치로 매핑하는 할고리즘과 자료구조를 고안해야 한다.

- 파일 구조 **File structure**
  - 논리 저장 단위 **Logical storage unit**
  - 관련된 정보의 집합 **Collection of related information**

- 메모리와 디스크간이 입출력 전송은 블록단위**(**예**: 512**바이트**)**

- 파일 제어 블록 – 소유와 접근허가와 같은 파일에 관한 정보로 구성된 저장 구조
  File Control Block – storage structure consisting of information about a file

계층화된 파일시스템Layered File System



- 응용프로그램
- 논리 파일 시스템: 메타데이터 정보를 관리하며, 실제 파일내용에 관한 데이터를 제외한 디렉터리 구조를 관리하며 파일 제어블록을 통해 유지된다. (디렉토리 구조)
- 파일구성모듈: 사용중인 파일할당의 타입과 파일의 위치를 알고 있기 때문에 이 모듈은 논리 블록 주소를 물리 블록 주소로 변환할 수 있다.(파일의 위치)
- 기본파일시스템: 특정 장치드라이버에게 디스크 상의 물리 블록을 읽고 쓰도록 명령을 내리는 층(물리 블록)
- 입출력제어:장치드라이버와 인터럽트 핸들러로 구성. 메모리와 디스크 시스템간의 정보전송을 담당. 장치드라이버는 번역기로 장치 내의 어떤 위치에 무슨 작업을 해야할 지를 알린다.
- 장치

A Typical File Control Block

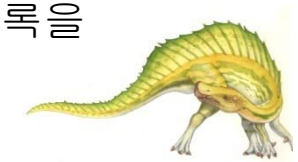
- **11.2** 파일 시스템 구현
  - 디스크상의 파일시스템: 디스크에 저장된 운영체제를 어떻게 부트시키는지, 블록의 총 개수, 가용 블록의 수와 위치, 디렉터리구조 그리고, 개별 파일에 대한 정보를가지고 있다.
  - 메모리 상의 파일시스템: 파일시스템관리와 캐싱을 통한 성능 향상을 이해 사용되며, 마운트시점에 적재되고, 파일시스템 동작중에 갱신되며, 마운트 해제 시에 제거된다.

- **FCB**
  - 파일의 정보

file permissions
file dates (create, access, write)
file owner, group, ACL
file size
file data blocks or pointers to file data blocks

<FCB 구조>

- 파일 접근권한
- 파일 생성, 접근, 쓰기 일자
- 파일 소유자
- 파일 크기
- 파일 데이터 블록이나 그 블록을 지시하는 포인터

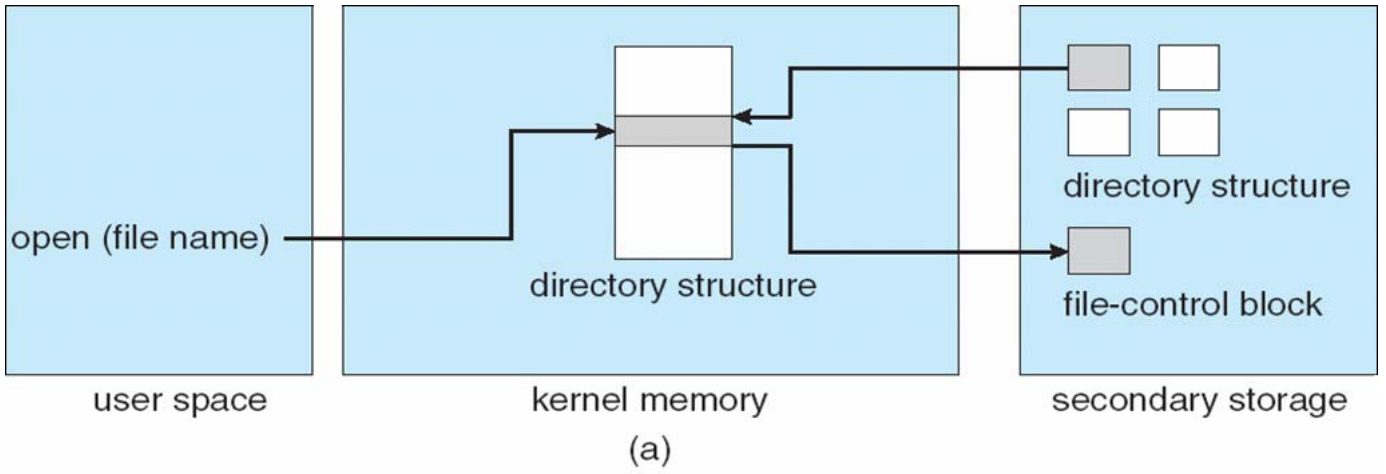


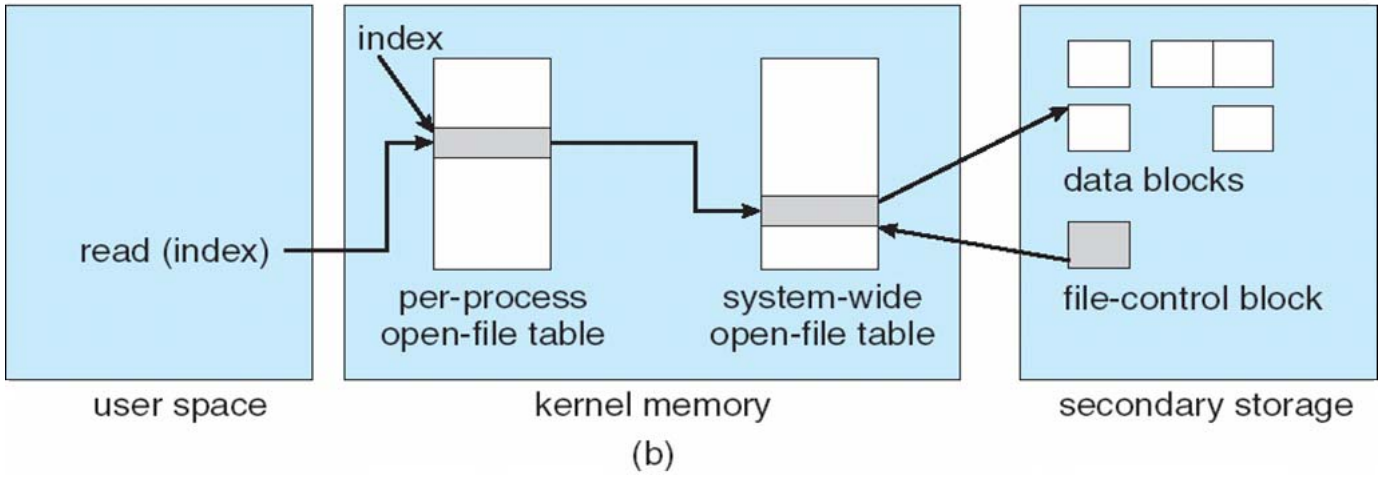
메모리 상주 파일 시스템 구조
In-Memory File System Structures

- 범시스템 열린 파일 테이블**(system wide open file table):** 모든 열린 파일의 **FCB**의 복사본을 가지고 있다**.**

- 프로세스별 열린 파일 테이블**(per-process open file table):** 범시스템 오픈 파일 테이블 내의 해당 항목에 대한 포인터를 포함한다**.**

In-Memory File System Structures





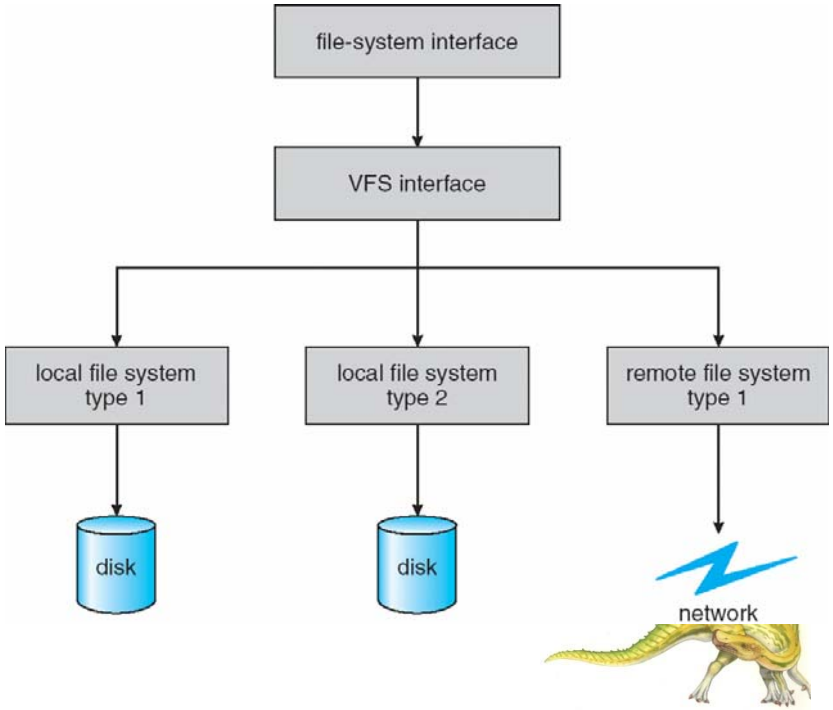
11.2.3 가상파일시스템
Virtual File Systems

- **VFS란?**
  - 일반적으로 구현을 단순화하여 조직화하고, 모듈화하기 위해 가상파일시스템을 구현하고 파일시스템 구현의 객체지향적인 방법을 제공
  - 한다. 따라서 크게 다른 파일 시스템 타입이 같은 구조 안에 구현되는 것을 허용한다.

Schematic View of Virtual File System

- 가상파일시스템인터페이스:
  - **NFS**는 **VFS** 인터페이스를 명확하게 정의함으로써 파일 시스템의 일반적 연산을 구현과 분리시킨다.
  - **VFS**는 파일을 네트워크 전체에서 고유하게 표현할 수 있는 **vnode**라 불리는 파일 표현 기법을 제공한다.
  - **vnode :** 네트워크 전체에서 유일한 파일에 대한 지정자**(designator)**

- **VFS**인터페이스는 현재 다루고 있는 객체의 종류에 관심이 없고,
- 함수테이블로 부터 적절한 함수를 호출하여 해당연산 수행
- 다시 말해,
- **VFS**는 **inode**가 디스크파일, 디렉터리 파일, 원격파일인지 관심이 없다.
- 다만 **read()**와 같은 연산은 함수테이블의 해당위치에 존재하고 그 함수만 호출하는 역할을 한다.



11.3 디렉터리 구현
Directory Implementation

- 디렉터리 구현의 단순한 방법
  - 선형리스트 : 데이터블록들에 대한 포인터를 가진 파일 이름의 선형리스트
    - 프로그램하기 단순
    - 선형탐색으로 실행을 위한 시간 소모
      - 파일생성 시: 디렉터리 탐색→ 파일 존재확인 → 디렉터리의 끝부분에 항목을 첨가
      - 파일삭제 시: 디렉터리에서 이름 탐색→ 할당공간방출
  - 해시테이블 – 해시데이터구조를 가진 선형 리스트
    - 디렉터리 탐색시간을 감소시킨다
    - 충돌 – 두 파일 이름이 같은 위치로 해시되는 상황
    - 해시테이블의 고정된 크기로 인한 문제가 있다. **fixed size**
    - 해시함수의 제약에 대한 문제가 있다.

11.4 할당 방법 Allocation Methods

- 할당 방법은 디스크 블록들이 파일을 위해 할당되는 방법을 의미한다. **An allocation method refers to how disk blocks are allocated for files:**

- **Contiguous allocation**
  - 연속 할당
- **Linked allocation**
  - 연결 할당
- **Indexed allocation**
  - 인덱스 할당

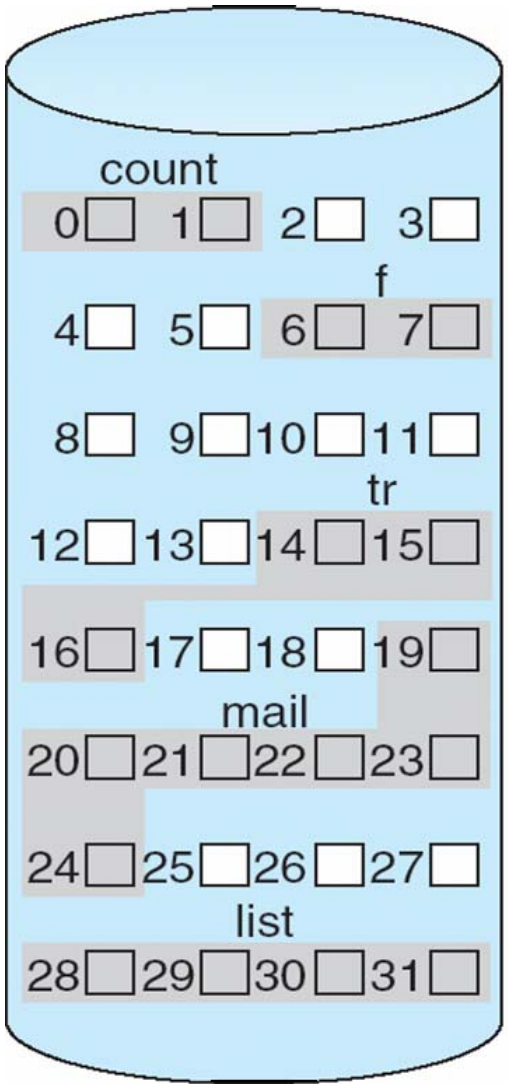
11.4.1 연속할당 Contiguous Allocation

- 각 파일은 디스크상에서 연속적인 블록들의 집합에 할당
  - 새로운 파일을 위한 가용공간 탐색문제가 있다.
  - 최초나 최적적합 할당으로 외부단편화문제가 있다.--> 압축

- 동적 연속할당으로 양 끝에 인접한 공간이 모두 사용 중이며 확장이 어렵다.
  - 해결책**1:** 생성불가, 프로그램 종료
  - 해결책**2:** 큰 가용공간으로 복사 후 기존공간 해제.

- 동적으로 파일 생성 시 전체 크기를 알 수 없어 임의의 사이즈를 할당하면서 공간 낭비 가능 **(**동적 저장장치 할당 문제**) Wasteful of space (dynamic storage-allocation problem)** → 일정한 크기 할당 후 **extent**로 할당

Contiguous Allocation of Disk Space



directory

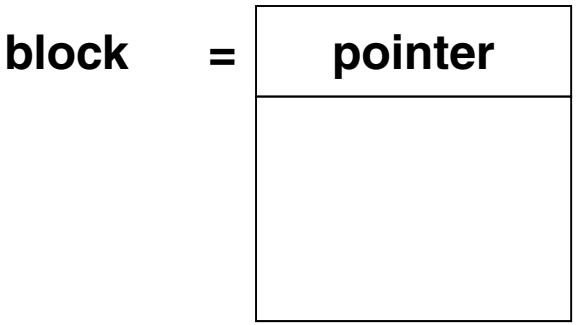
file	start	length
count	0	2
tr	14	3
mail	19	6
list	28	4
f	6	2

확장 기반의 시스템 Extent-Based Systems

- **extent-based system**
  - 수정된 연속할당
  - **extent 단위로** 디스크블록들을 할당
- **extent**
  - 디스크의 연속 블록 할당
  - **extent**는 파일 할당을 위해 할당
  - 파일은 하나이상의 **extent**들로 구성
- **extent** 의 크기에 따라 내부/외부단편화문제가 발생가능하다.

11.4.2 연결 할당 Linked Allocation

- 각 파일은 디스크 블록의 연결리스트
- 디렉터리는 파일의 시작과 마지막 블록을 가르키는 포인터를 가지고 있다.

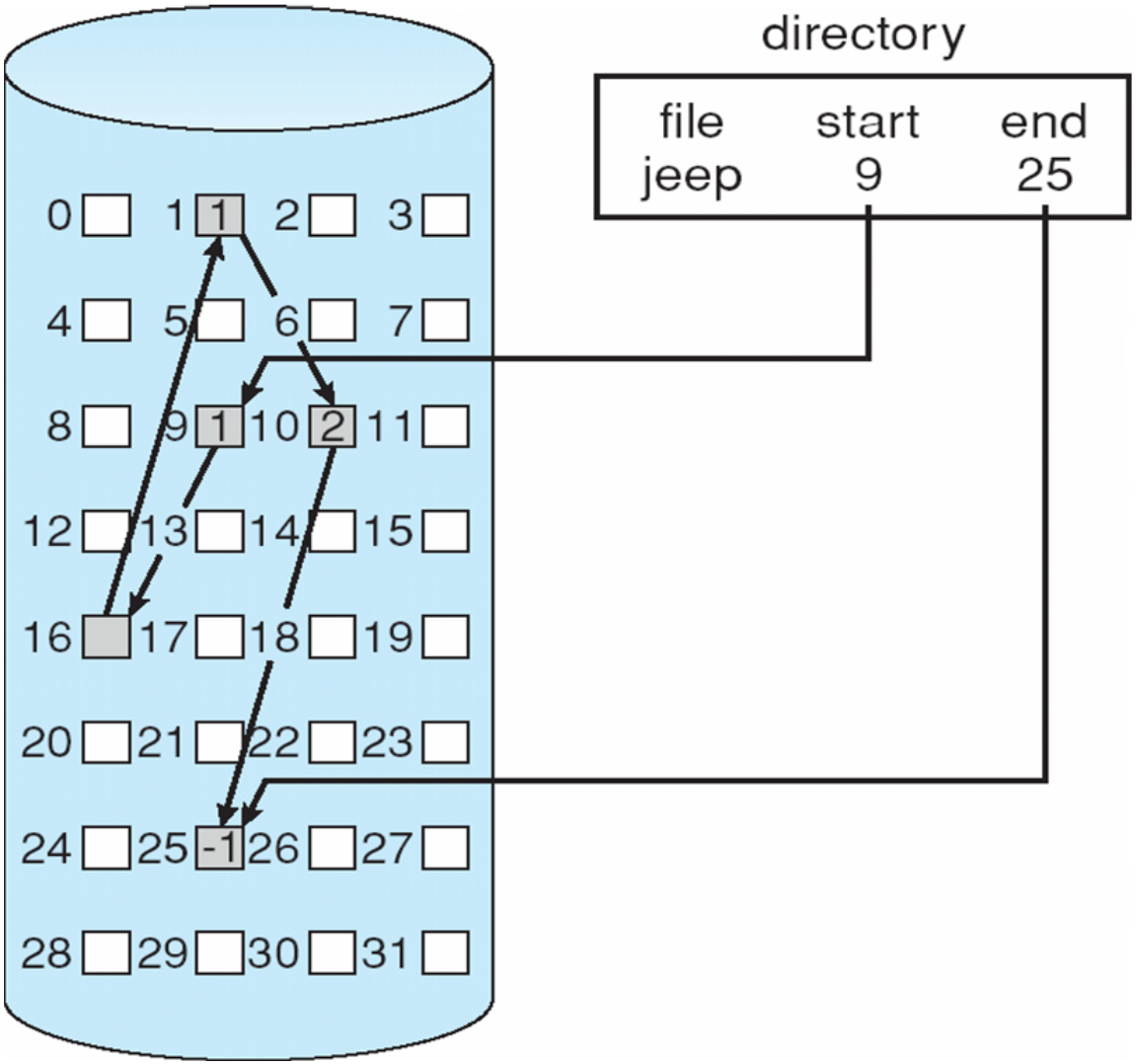


연결 할당 Linked Allocation (Cont.)

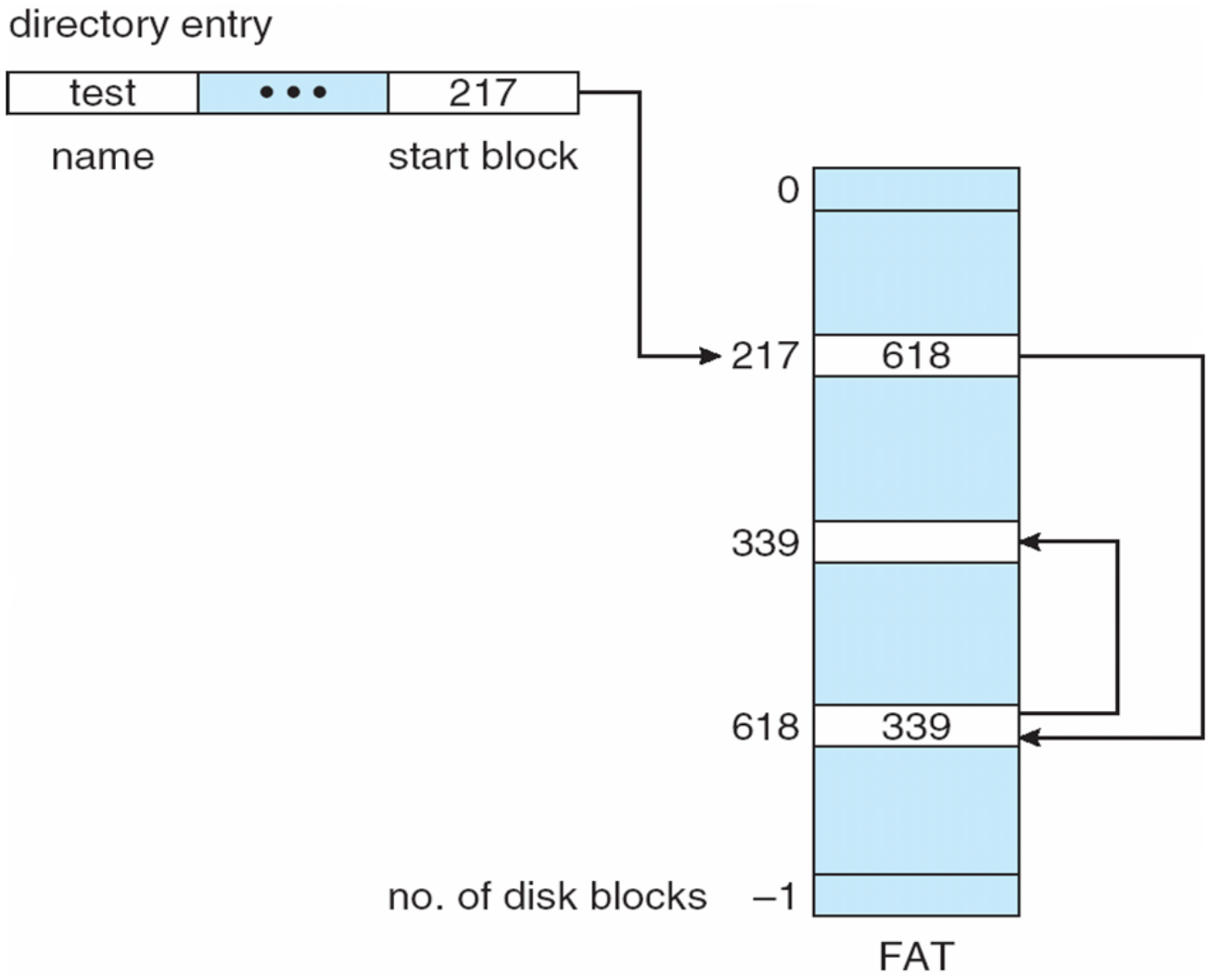
- 장점
  - 시작주소만 필요할 만큼 단순
  - 자유공간 관리 시스템 – 공간의 낭비가 없다
- 단점
  - 임의 접근이 안됨. **No random access**

파일 할당 테이블**(FAT)- MS-DOS** 나 **OS/2**에서 사용된 디스크공간 할당

Linked Allocation



File-Allocation Table



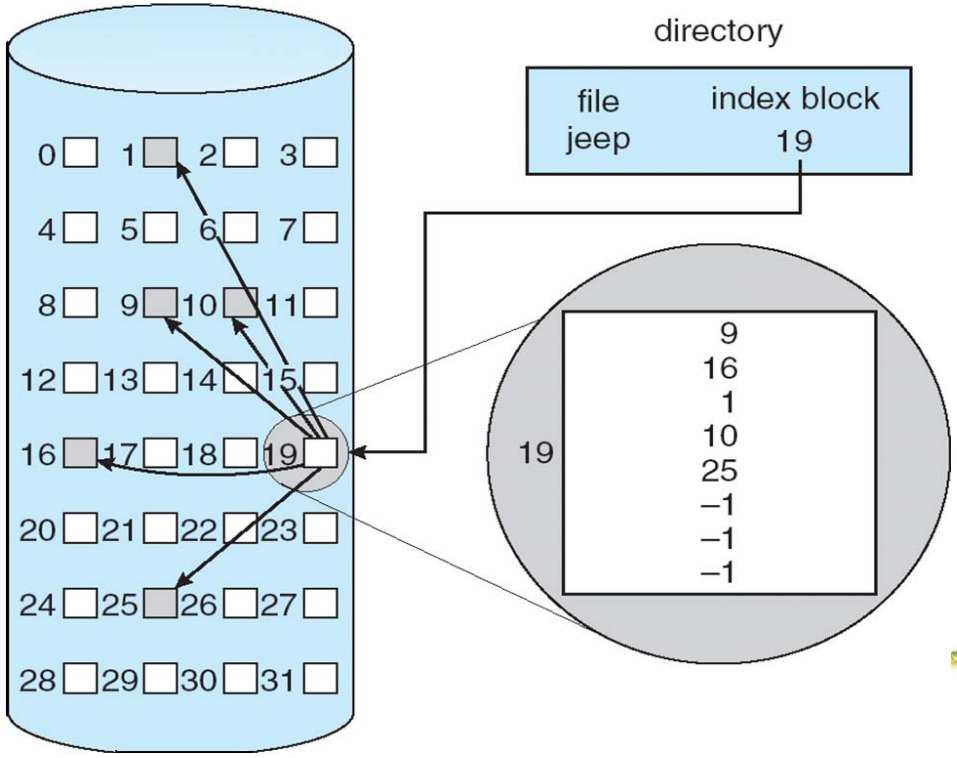
11.4.3 색인 할당 Indexed Allocation

- **Index Allocation**의 정의
  - **Linked Allocation**은 연속할당의 외부 단편화 문제와 파일 크기 선언 문제를 해결
  - **FAT**가 없다면 연결할당은 직접 접근을 효율적으로 지원할 수 없다**.** 따라서 색인 할당은 모든 포인터를 한군데**,** 즉 색인블록으로 모아 놓음으로써 이 문제를 해결

- 모든 포인터를 인덱스블록으로 가져온다

Example of Indexed Allocation

- 장점
  - 임의접근이 가능 **Random access**
- 단점
  - 외부단편화 없이 동적 접근이 가능하지만, 색인을 위한 공간의 낭비가 문제점로 인덱스 블록의 포인터 오버헤드는 연결할당의 포인터 오버헤드보다 크다.



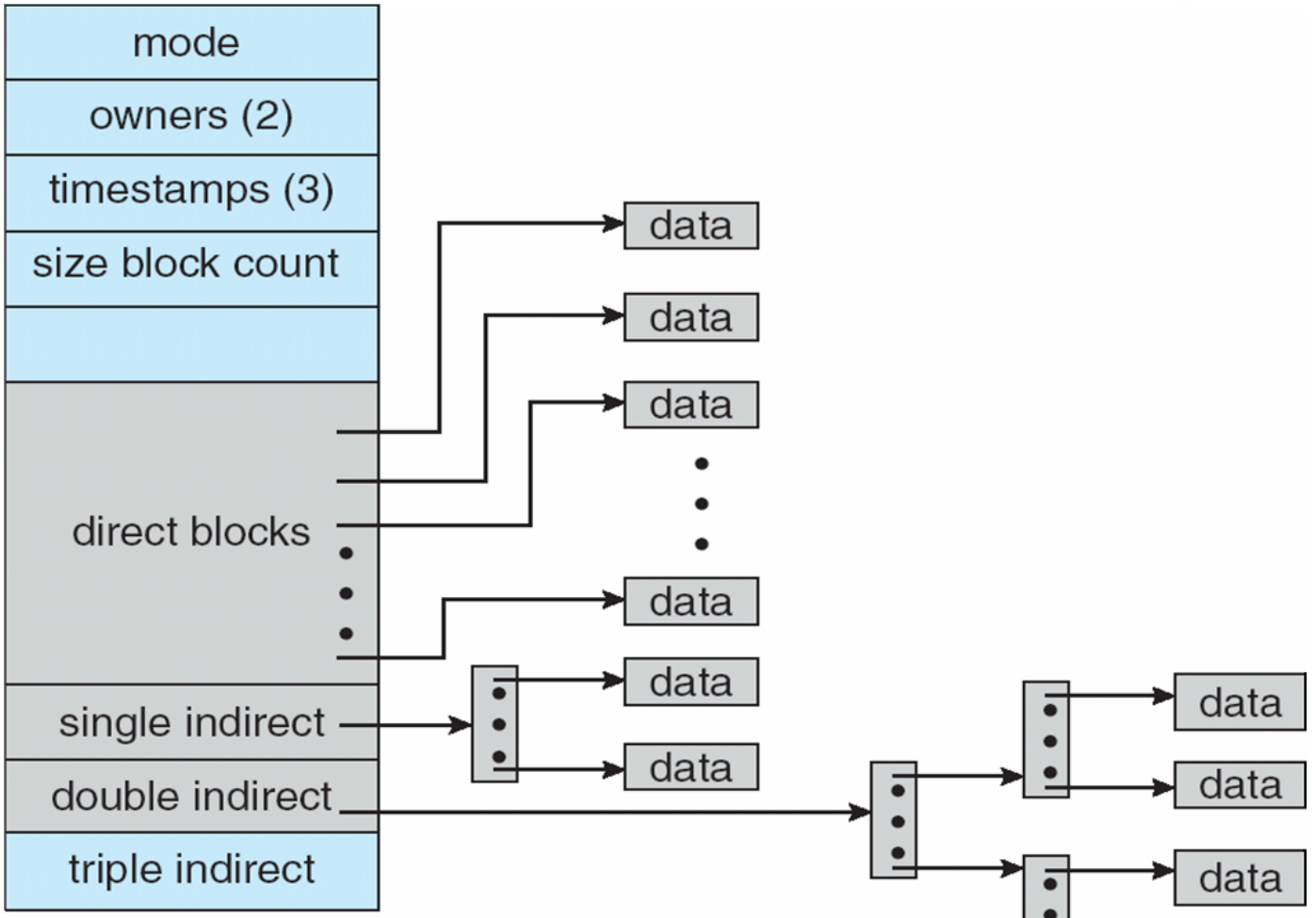
Indexed Allocation – Mapping (Cont.)

- 길이 제안이 없는 파일에서 논리에서 물리로 매핑 **Mapping from logical to physical in a file of unbounded length (block size of 512 words).**

- 색인할당의 블록크기 해결책
  - 연결기법 **-** 인텍스 테이블의 색인 블록을 연결리스트로 연결
  - 다중수준인덱스**-** 일 단계 인덱스 블록이 **2**단계 색인블록의 집합을 가르키고, 이 단계 색인 블록이 파일 블록을 가르키게 하는 방식 **Two-level index (maximum file size is 512^3)**

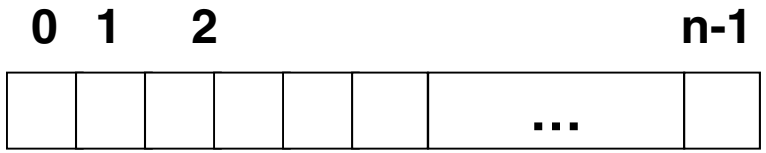
- 결합기법 **:** 직접블록과 간접블록을 구분한 할당방법**. 15**개의 인덱스를 파일의 인덱스에 유지하고**, 12**개는 직접접근을 위해 사용하고**, 3**개의 포인터는 간접접근을 위해 사용한다**.** 간접은 또 다른 직접과 간접으로 분할**.** 간접블록**=**단일 간접**+** 이중간접**.** 이중간접**=**단일이중간접**+**삼중간접

Combined Scheme: UNIX (4K bytes per block)



11.5 자유공간관리
Free-Space Management

- **Bit vector (*n* blocks):** 프리면 **1,** 사용중이면 **0**



$$\text{bit}[i] = \begin{cases} 0 \Rightarrow \text{block}[i] \text{ free} \\ 1 \Rightarrow \text{block}[i] \text{ occupied} \end{cases}$$

Block number calculation=워드의비트수 * 값이**0**인워드수 **+** 첫번째**1**비트변위

**(number of bits per word) *
(number of 0-value words) +
offset of first 1 bit**

Free-Space Management (Cont.)

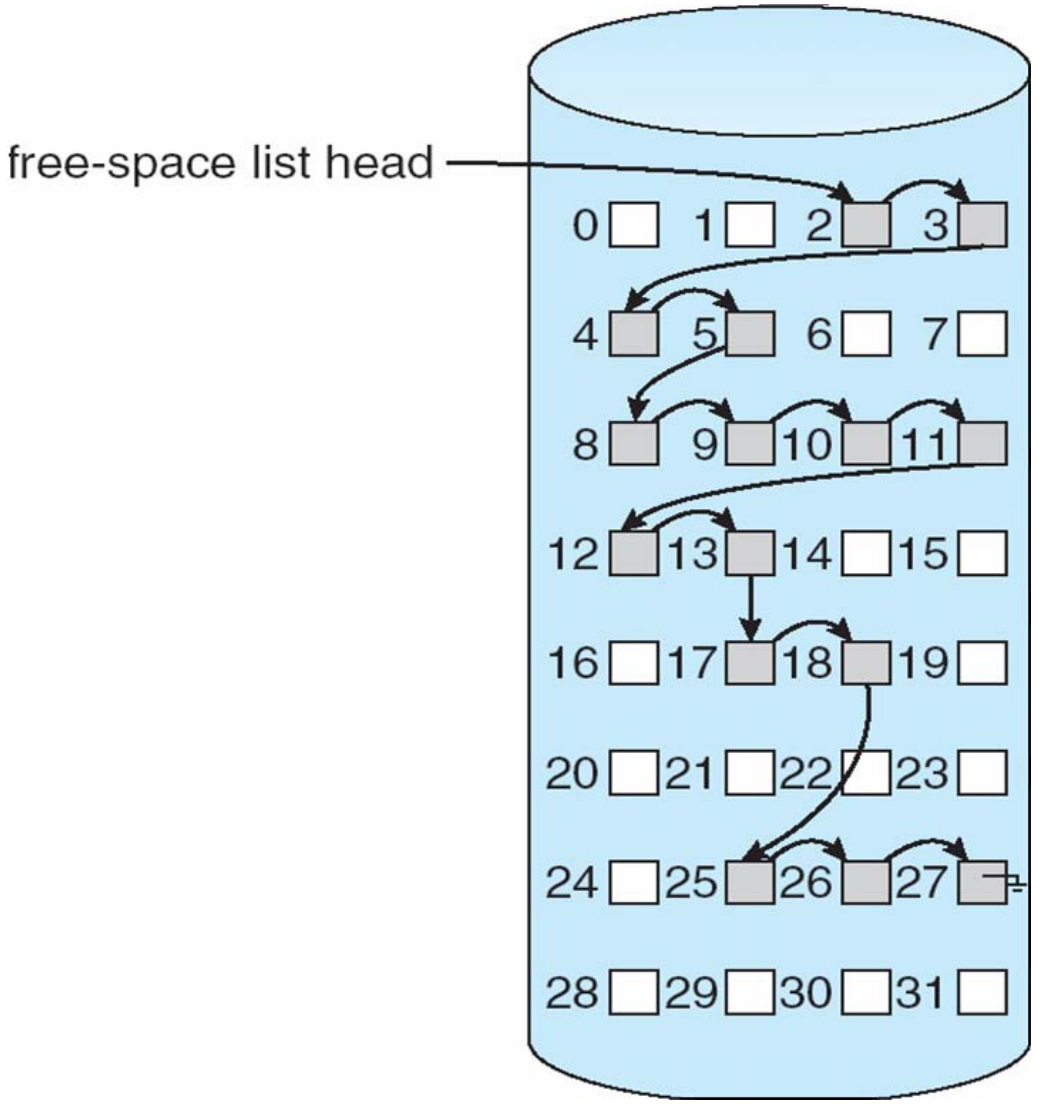
- **Bit Vector**
  - 비트 벡터를 유지하기 위한 여분의 공간을 필요
    - **Example:**

      block size = 2^{12} bytes

      disk size = 2^{30} bytes (1 gigabyte)

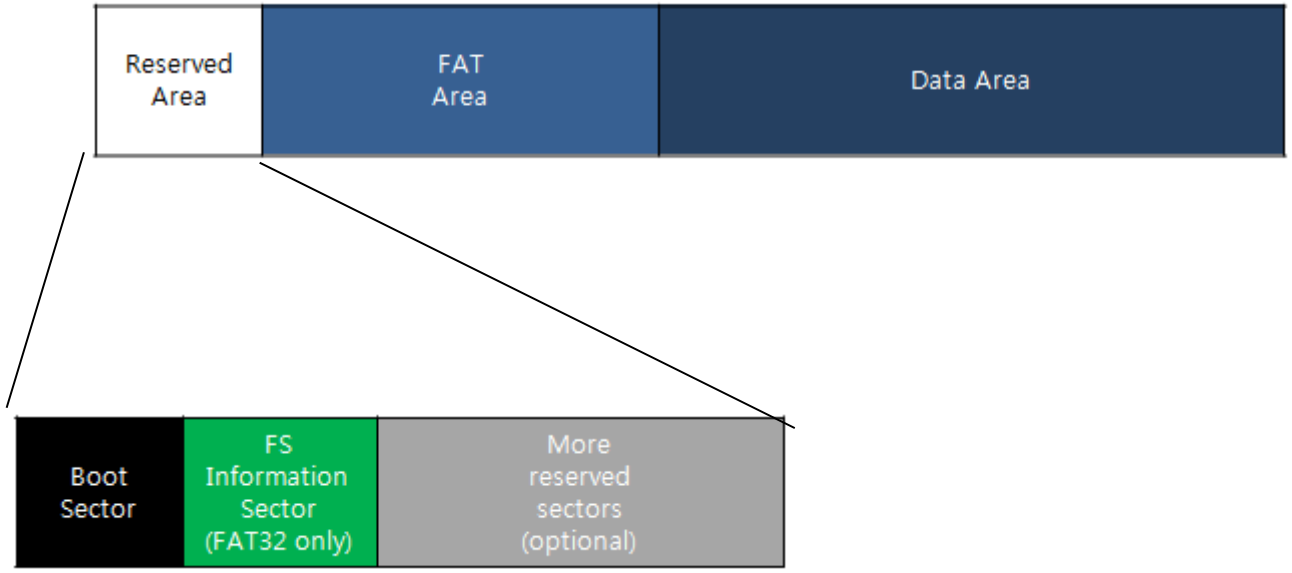
      n **= $2^{30}/2^{12} = 2^{18}$ bits (or 32K bytes)**
  - 장점 : 연속 파일을 확보하기 쉬움
  - 단점 : 디스크 크기가 커지면 비트맵에 할당하는 요구량이 커지게 된다.
- 링크드 리스트 **Linked list (free list)**
  - 장점 : 연속공간을 쉽게 얻을 수 없다
  - 단점 : 공간 낭비가 없다
- 그룹핑 **Grouping**
  - 빈공간의 첫 시작블록에 **n**개의 블록주소를 저장하는 방법
- 계수 **Counting**
  - 가용블록의 첫번째 주소와 연속된 **n**개의 블록개수를 저장하는 방법

Linked Free Space List on Disk



FAT

- **DISK Structure**



FAT

	Sector Number	Sector Number (Hex)	What it does
Boot Track	0	0	contains information about the disk, plus code that's run by the BIOS when the disk is started up.
1st File Allocation Table (FAT)	1 - 9	1 - 9	Information on what clusters are allocated on the disk.
2nd File Allocation Table (FAT)	10 - 18	0Ax – 12x	Redundant information on what clusters are allocated on the disk.
Root Disk Directory	19 - 32	13X – 20X	Information about the characteristics of the directories and files.
File / Directory Data Area	33 - 2879	21X - B3Fx	The contents of the files and Directories

FAT

ROOT DIRECTORY

File Name	File Extension	Date	Time	File Size	Starting Cluster
MYFILE	TXT	1/23/98	11:13 PM	41,364	1
YOUR	XLS	11/7/99	1:00 AM	98,509	3
THIS	DOC	5/7/00	2:13 AM	38,949	7

FAT

Cluster	Pointer	Data in Cluster
1	2	MYFILE.TXT
2	EOF	MYFILE.TXT
3	4	YOUR.XLS
4	5	YOUR.XLS
5	6	YOUR.XLS
6	EOF	YOUR.XLS
7	8	THIS.DOC
8	EOF	THIS.DOC
9	0	Unused
10	BAD	Contains bad sectors

www.cod.edu/people/faculty/houhain

Cluster Chain in FAT

xxxxxxxx	xxxxxxxx	00000009	00000004
00000005	00000007	00000000	00000008
FFFFFFFF	0000000A	0000000B	00000011
0000000D	0000000E	FFFFFFFF	00000010
00000012	FFFFFFFF	00000013	00000014
00000015	00000016	FFFFFFFF	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000
00000000	00000000	00000000	00000000

Root Directory:
2, 9, A, B, 11

File #1:
3, 4, 5, 7, 8

File #2:
C, D, E

File #3:
F, 10, 12, 13, 14, 15, 16

Log Structured File Systems

- 로그구조 파일 시스템은 보편화된 기술로 각각의 업데이트를 **트랜잭션**으로 파일시스템에 기록한다

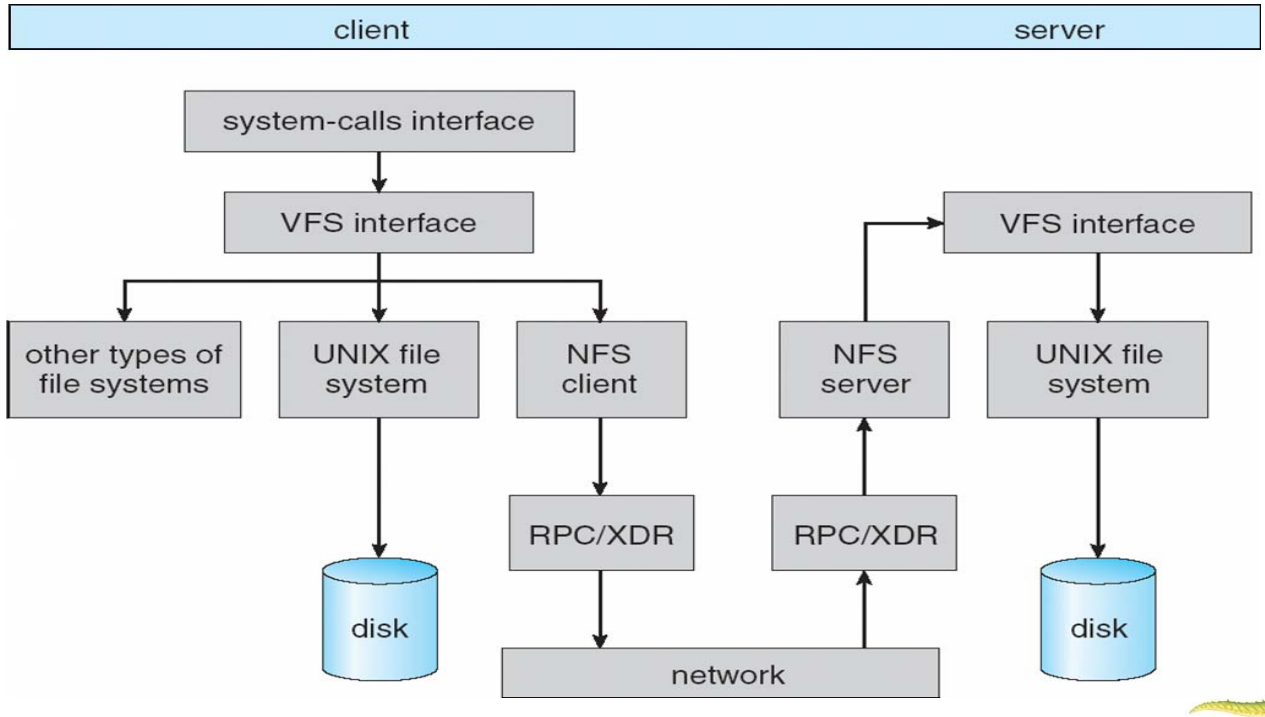
- 모든 트랜잭션은 로그에 기록된다
  - 트랜잭션은 로고에 기록되면 허가로 간주된다
  - 그러나, 파일시스템은 갱신되지 않을 수 도 있다.
- 로그의 트랜잭션은 파일시스템 동기적으로 기록된다.
  - 파일시스템이 수정될 때 트랜잭션은 로그로부터 제거된다.
  -
- 만일 파일시스템이 파괴되면 로그에 있는 모든 남아있는 트랜잭션은 여전히 수행된다.

- 트랜잭션이 중단된 경우, 즉 시스템이 확약(허가, **committed**)되지 않은 트랜잭션은 원상복구

11.8 Network File System (NFS)

- **NSF**는 랜을 통한 원격지 파일을 접근하기 위한 소프트웨어 시스템의 구현과 명세를 모두 말한다.

- **NFS**의 목적은 일정수준의 공유를 투명하게 허용하고, **C/S** 모델을 기반으로 한다.
- 투명성(원격시스템의 파일과 디렉터리를 로컬의 파일시스템에 존재하는 것처럼 접근하는 것)



APPENDIX

인용: web.cs.wpi.edu/~jb/

FAT 12

Sector 001x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	0123456789ABCDEF0123456789ABCDEF
00000000	F0	FF	FF	FF	FF	FF	FF	6F	00	07	80	00	09	A0	00	0B	F0	FF	FF	0F	00	00	00	00	00	00	00	00	00	00	00	00	.....o.....
00000020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000200																																	

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
F0	FF	FF	FF	FF	FF	FF	6F	00	07	80	00	09	A0	00	0B	F0	FF	FF	0F	00

Cluster 2 3 4
These files have a single sector

FF FF Padding

F0 means it's a floppy.

Cluster 5: Aims at Cluster 6

Cluster 6: Aims at Cluster 7

Cluster 7: Aims at Cluster 8

Cluster 8: Aims at Cluster 9

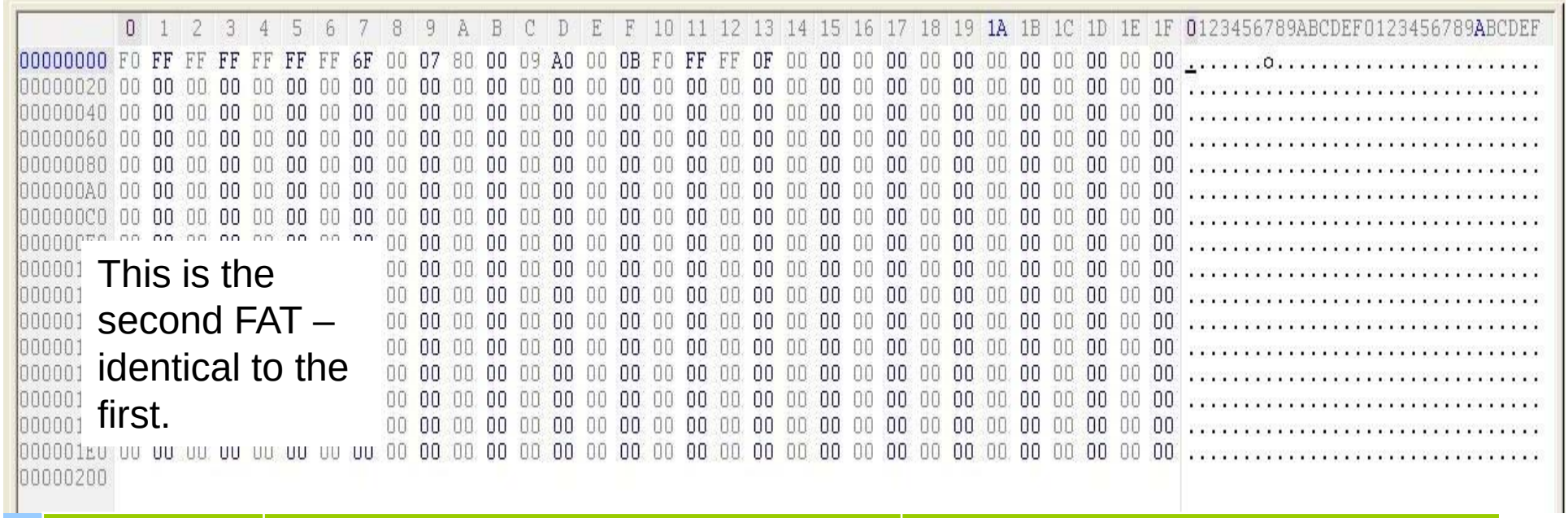
Cluster 9: Aims at Cluster A

Cluster A: Aims at Cluster B

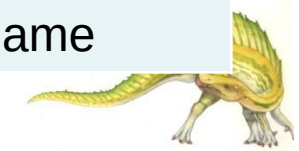
Cluster B: Last Sector (FFF)

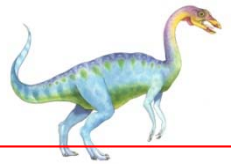
Cluster C: Last Sector (FFF)

Sector 00Ax



Starting Cluster	Contains (Points to)	What does it hold?
2	FFF (one sector only)	(021x) A_Dir -- Directory
3	FFF (one sector only)	(022x) File in directory A_Dir
4	FFF (one sector only)	(023x) File1 (single sector)
5	6 → 7 → 8 → 9 → A → B → FFF	(024x-02Ax) Multisector file
C	FFF (one sector only)	(02Bx) File with long name



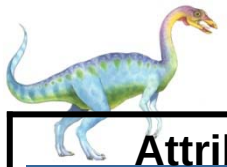


FAT 12 Directory Information

Directory Entries hold information about the files or directories that are in that directory. Below is the format for a directory entry.

Attribute	Offset	Number of bytes in field
Filename	0	8
Filename Extension	8	3
File Attributes	B	1
reserved	C	10
Creation Time	16	2
Creation Date	18	2
Pointer to cluster in FAT	1A	2
File Length	1C	4

See the next page for additional information about many of these fields.



FAT 12 Directory Information

Attribute	Offset	Number of bytes in field
Filename	0	8
Filename Extension	8	3
File Attributes	B	1
reserved	C	10
Creation Time	16	2
Creation Date	18	2
Pointer to cluster in FAT	1A	2
File Length	1C	4

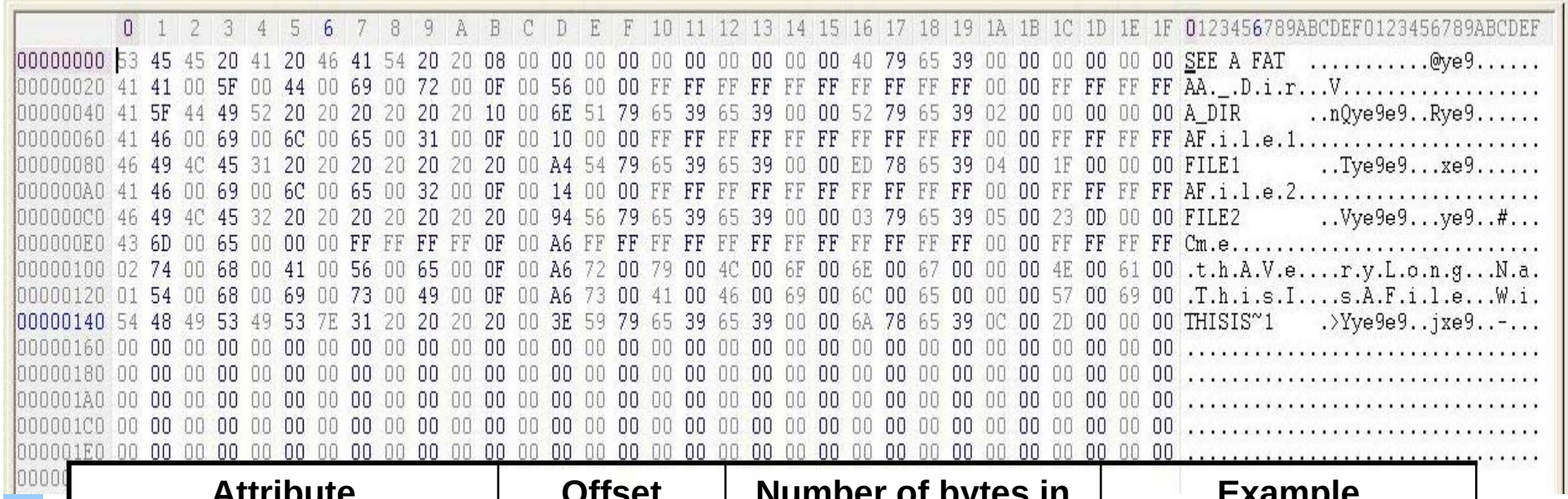
First byte of filename:
- **$00 – unused**
- **$E5 – file erased**
- **$2E – subdirectory file**

Attributes:
- **bit 0 – read only**
- **bit 1 – hidden file**
- **bit 2 – system file**
- **bit 3 – volume label**
- **bit 4 – subdirectory**
- **bit 5 – archive bit**
- **bits 6, 7 -- unused**

Creation Time:
- **bits 0 – 4 – seconds/2 (0 – 29)**
- **bits 5 – 10 – minutes (0 – 59)**
- **bits 11 – 15 – hours (0 – 23)**

Creation Date:
- **bits 0 – 4 – date (1 – 31)**
- **bits 5 – 8 – month (1 - 12)**
- **bits 9 – 15 – year since 1980**

Root Directory FAT 12 Sector 013x



	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	0123456789ABCDEF0123456789ABCDEF
00000000	53	45	45	20	41	20	46	41	54	20	20	08	00	00	00	00	00	00	00	00	00	00	40	79	65	39	00	00	00	00	00	00	SEE A FAT .....@ye9.....
00000020	41	41	00	5F	00	44	00	69	00	72	00	0F	00	56	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	AA._.D.i.r...V.....
00000040	41	5F	44	49	52	20	20	20	20	20	20	10	00	6E	51	79	65	39	65	39	00	00	52	79	65	39	02	00	00	00	00	00	A_DIR ..nQye9e9..Rye9.....
00000060	41	46	00	69	00	6C	00	65	00	31	00	0F	00	10	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	AF.i.l.e.1.....
00000080	46	49	4C	45	31	20	20	20	20	20	20	20	00	A4	54	79	65	39	65	39	00	00	ED	78	65	39	04	00	1F	00	00	00	FILE1 ..Tye9e9...xe9.....
000000A0	41	46	00	69	00	6C	00	65	00	32	00	0F	00	14	00	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	AF.i.l.e.2.....
000000C0	46	49	4C	45	32	20	20	20	20	20	20	20	00	94	56	79	65	39	65	39	00	00	03	79	65	39	05	00	23	0D	00	00	FILE2 ..Vye9e9...ye9..#...
000000E0	43	6D	00	65	00	00	00	FF	FF	FF	FF	0F	00	A6	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	Cm.e.....
00000100	02	74	00	68	00	41	00	56	00	65	00	0F	00	A6	72	00	79	00	4C	00	6F	00	6E	00	67	00	00	00	4E	00	61	00	.t.h.A.V.e....r.y.L.o.n.g...N.a.
00000120	01	54	00	68	00	69	00	73	00	49	00	0F	00	A6	73	00	41	00	46	00	69	00	6C	00	65	00	00	00	57	00	69	00	.T.h.i.s.I....s.A.F.i.l.e...W.i.
00000140	54	48	49	53	49	53	7E	31	20	20	20	20	00	3E	59	79	65	39	65	39	00	00	6A	78	65	39	0C	00	2D	00	00	00	THISIS~1 .>Yye9e9..jxe9..-...
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0000																																	

Attribute	Offset	Number of bytes in field	Example
Filename	0	8	File2
Filename Extension	8	3	---
File Attributes	B	1	20
reserved	C	10	
Creation Time	16	2	“03 79”
Creation Date	18	2	“65 39”
Pointer to cluster in FAT	1A	2	5
File Length	1C	4	0D23

Sector 021x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	0123456789ABCDEF0123456789ABCDEF
00000000	2E	20	20	20	20	20	20	20	20	20	20	10	00	6E	51	79	65	39	65	39	00	00	52	79	65	39	02	00	00	00	00	00	. ..nQye9e9..Rye9.....
00000020	2E	2E	20	20	20	20	20	20	20	20	20	10	00	6E	51	79	65	39	65	39	00	00	52	79	65	39	00	00	00	00	00	00	.. ..nQye9e9..Rye9.....
00000040	41	46	00	69	00	6C	00	65	00	49	00	0F	00	EA	6E	00	53	00	75	00	62	00	64	00	69	00	00	00	72	00	00	00	AF.i.l.e.I....n.S.u.b.d.i...r...
00000060	46	49	4C	45	49	4E	7E	31	20	20	20	20	00	99	51	79	65	39	65	39	00	00	27	79	65	39	03	00	27	00	00	00	FILEIN~1 ..Qye9e9..'ye9..'...
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	....
00000200																																	

This contains the contents of the subdirectory, A_Dir. Note that the format of a file entry is the same as for the root directory.

Sector 022x

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	0123456789ABCDEF0123456789ABCDEF
00000000	54	68	69	73	20	69	73	20	74	68	65	20	66	69	6C	65	20	69	6E	20	74	68	65	20	73	75	62	64	69	72	65	63	This is the file in the subdirec
00000020	74	6F	72	79	2E	0D	0A	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	tory.....
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000120	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000180	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
000001E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
00000200																																	

This contains the contents of the file AFileInTheSubdirectory. Notice that it's ALL data – no information about the file itself is kept in the data sectors. There is no reason for the OS to interpret this sector in any fashion.