

Hệ Điều Hành

(*Nguyên lý các hệ điều hành*)

Đỗ Quốc Huy

huydq@soict.hust.edu.vn

Bộ môn Khoa Học Máy Tính

Viện Công Nghệ Thông Tin và Truyền Thông

Chapter 4 file system management

- To keep information for long time and later use -> store on external memory (disk, magnetic tape, optical disk,...) => **file**
 - Data or program file
 - Many files ⇒ file system
 - Files system combined of 2 parts
 - files: Contain data/program
 - directory : provide information about file
- Files system is large ⇒ How to manage?
 - File's properties, required operation?
- How to store and access file on storage devices?
 - Storage space allocation, free memory management

an abstract datatype (implement in ≠ sys is ≠)

Chapter 4 File system management

- ① File system
- ② File system's implementation
- ③ Information organization on disk
- ④ FAT system

Chapter 4: File system management

1. File system

1.1 File's notion

- File's notion

- Directory structure

Chapter 4: File system management

1. File system

1.1 File's notion

Introduction

- Information is stored on different medias/devices
 - Example: Disk, tape, optical disk...
 - Storage device is modeled as an array of memory block



We do NOT count how many Bytes / bits – we count nb. of mem block / frame

- File is a set of information stored on storage devices
 - File is a storage unit of OS on external devices
 - File contains sequence of bits, bytes, lines, records,... Contain meaning defined by creator
- Structure of file is defined by type of file
 - Text file: Sequence of character organized into lines
 - Object file: Bytes organized into block to be read and edited by the linker
 - Executive File: Sequence of codes can be run in memory
 - ...

Unit of mem : cluster / sector ...

Chapter 4: File system management

1. File system

1.1 File's notion

File's attributes

- **File name:** sequence of character (e.g. “hello.c”)
 - Information store in the format that is readable by user
 - Can be distinguished by upper-case/lower-case
 - Guarantee the independence of file from process, user...
 - A created file hello.c by notepad on Windows
 - B uses emacs on Linux to modify a file specified by name hello.c
- **Identifier:** A tag to define an unique file
- **Type:** Used in system supports many types of file
 - Type of file is defined based on one part of file name
 - Example: .exe, .com/ .doc, .txt/ .c, .jav, .pas/ .pdf, .jpg,...
 - Based on type, OS decides corresponding operation
 - Run an execution file that source is modified ⇒ Recompile
 - Double click on a text file (*.doc) ⇒ Call word processor
- **Position:** Point to device and location of file on that device
- **Size:** Current/maximum size of file
- **Protection:** Control access: who can read/write...
- **Time:** Creation time, modified time, last used time ...

Difference between size of file vs. size on disk

Assume: file size: 16 KB ; memory \rightarrow clusters, 64 KB each
To save the file, we need at least 1 cluster for it

$\Rightarrow \left\{ \begin{array}{l} \text{size of file} = 16 \text{ KB} \\ \text{size on disk} : 64 \text{ KB} \end{array} \right.$

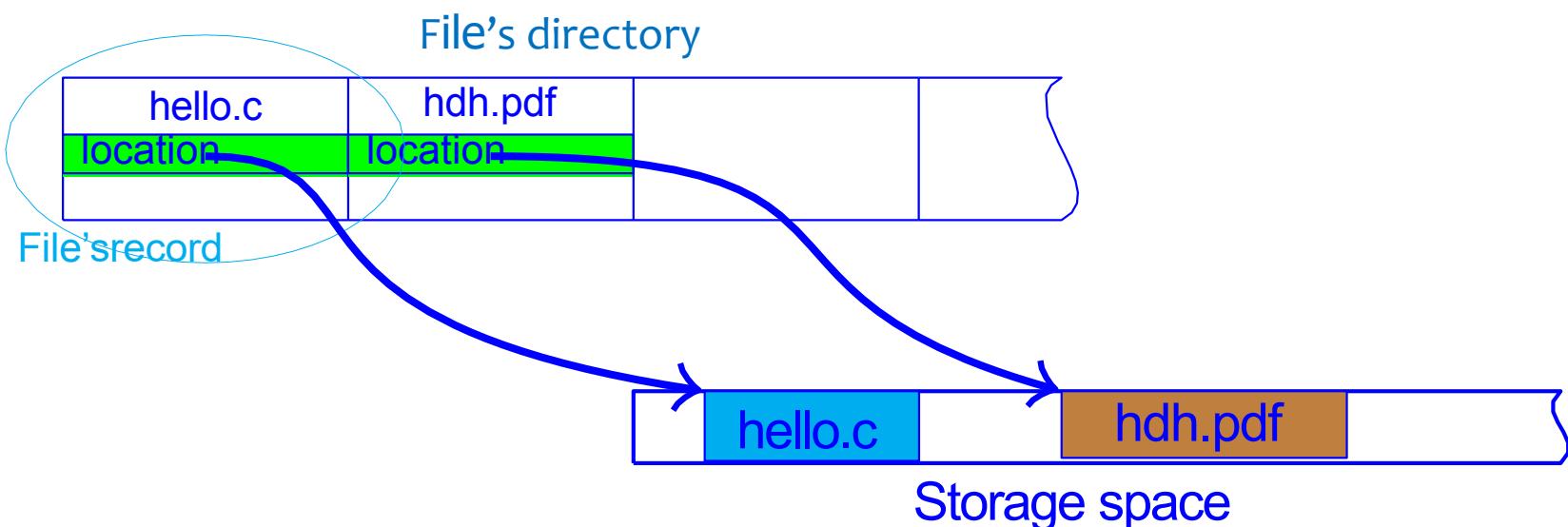
Chapter 4: File system management

1. File system

1.1 File's notion

File's attributes (cont.)

- File's attributes are stored in data structure: **File's record**
 - May contain only file's name and file's identifier; file's identifier define other information
 - Size from several bytes to kilobytes
- File's record are stored in **File's directory**
 - Size may be up to Megabytes
 - Often stored on external memory
 - Directory parts are loaded into memory when necessary



Chapter 4: File system management

1. File system

1.2. Directory structure

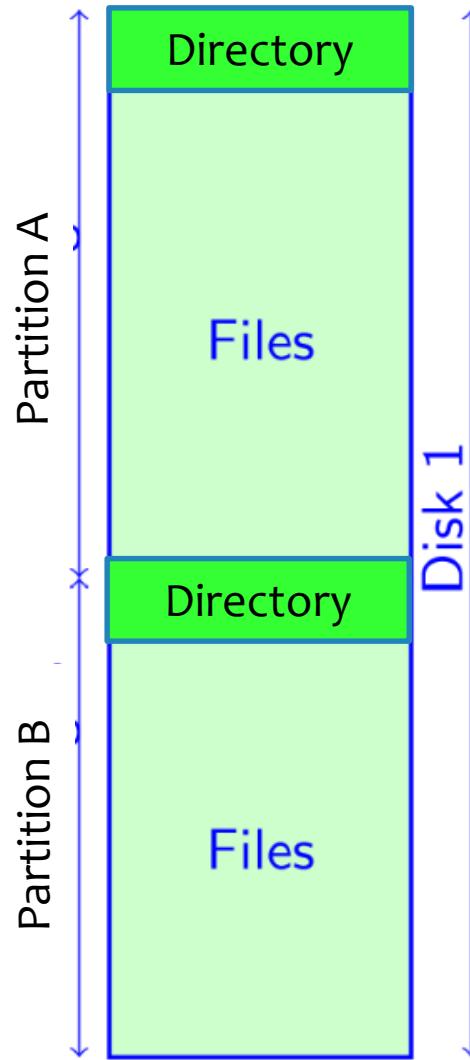
- File's notion
- Directory structure

Chapter 4: File system management

1. File system

1.2. Directory structure

Partition



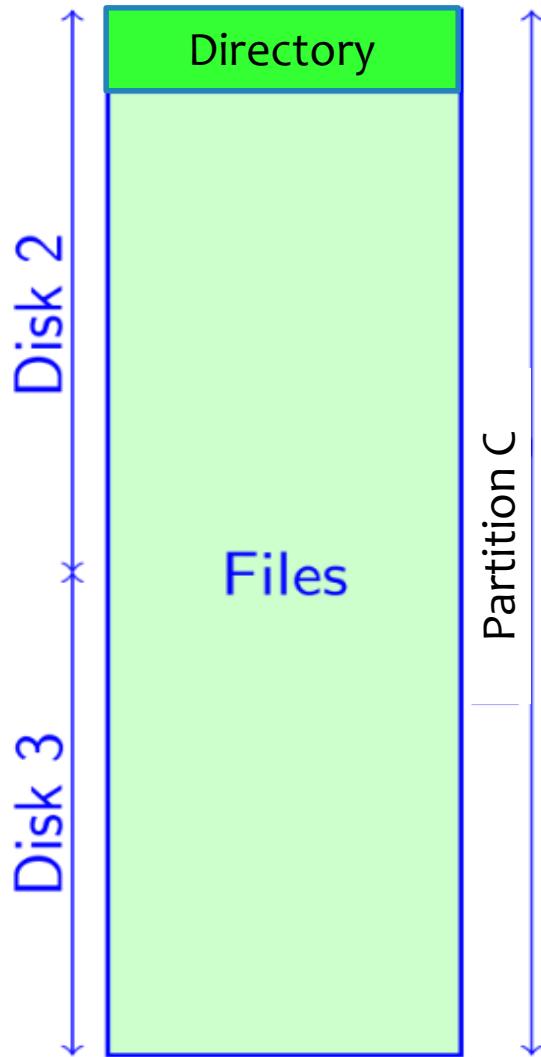
- Disk is divided into **Partitions, Minidisks, Volumes**
- Each partition is processed as an independent storage area
- Can store an individual OS

Chapter 4: File system management

1. File system

1.2. Directory structure

Partition



also possible

Merge several disks into a large logic structure

- User only care about file and directory's structure
- Do not care about how physical memory space is allocated to files

Chapter 4: File system management

1. File system

1.2. Directory structure

Operations with directory

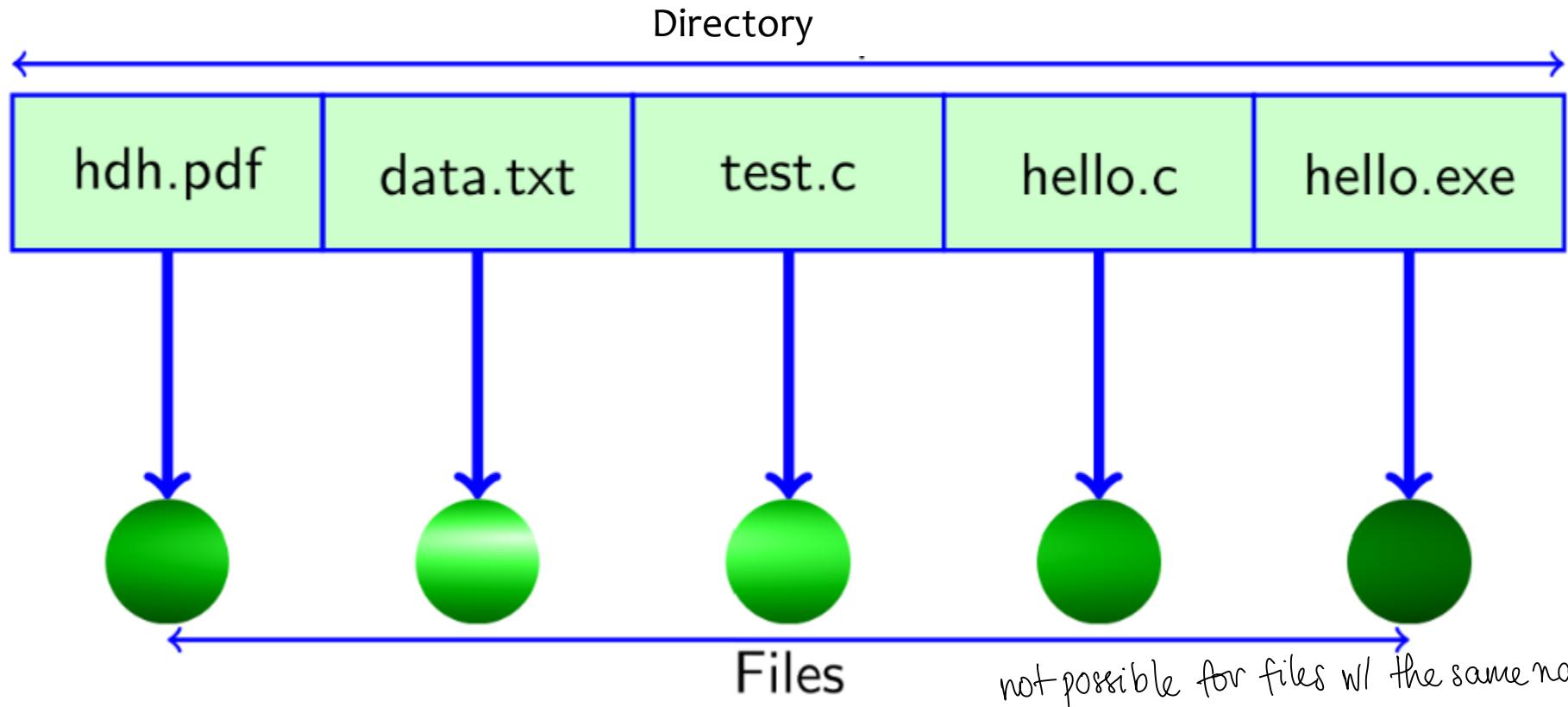
- Each partition contain information about contained files
 - File's information is stored in device's directory
- Directory is a **translation table** allow mapping from one name (**file**) to a member inside directories
 - Directory can be implemented by different methods
- Require operations for insert, create, delete or show the list
- Operations
 - **Seek file**: Search for an item corresponding to a file name
 - **Create file**: Create new file require to create new item in directory
 - **Delete file**: When delete a file, remove corresponding item in the directory
 - **Listing file**: Show the list of files and corresponding item in directory
 - **Rename file**: Rename file, position in the directory's structure
 - **Traverse the file system**: Access all directory and content of all files in the directory (**backup data to tape**)

Chapter 4: File system management

1. File system

1.2. Directory structure

One level directory



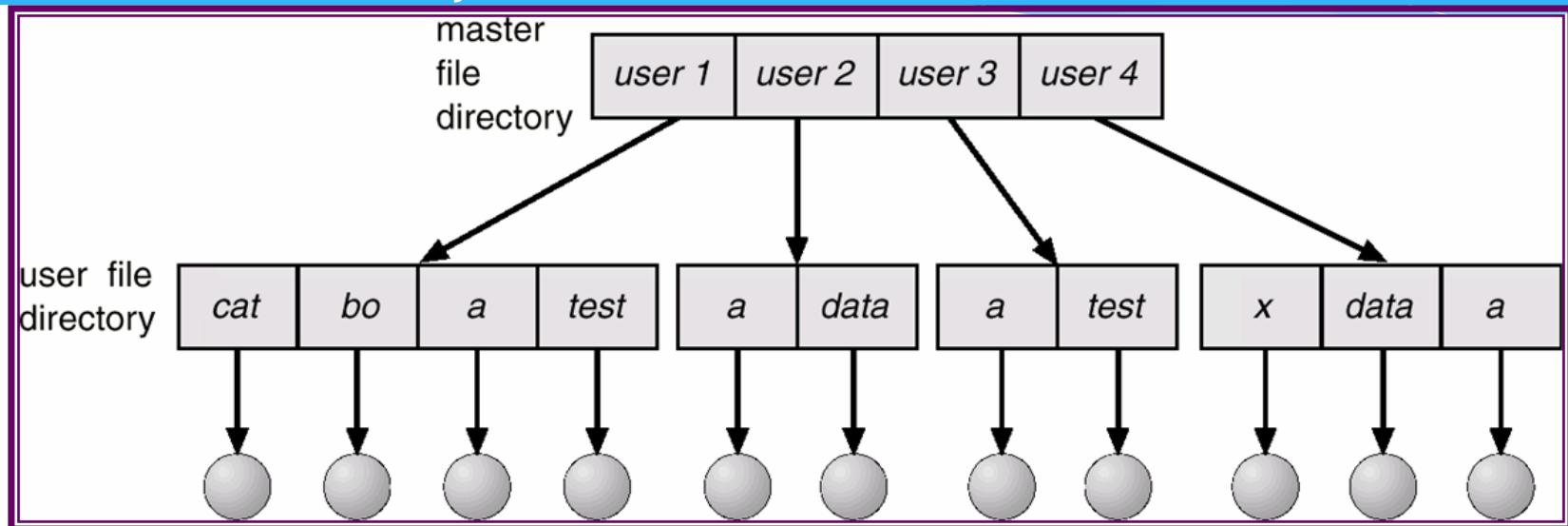
- Simplest structure, files are stored in the same directory
- If number of user and file is large, it's possible for the filename to be duplicated
 - Each user has his own directory riêng

Chapter 4: File system management

1. File system

1.2. Directory structure

One level directory



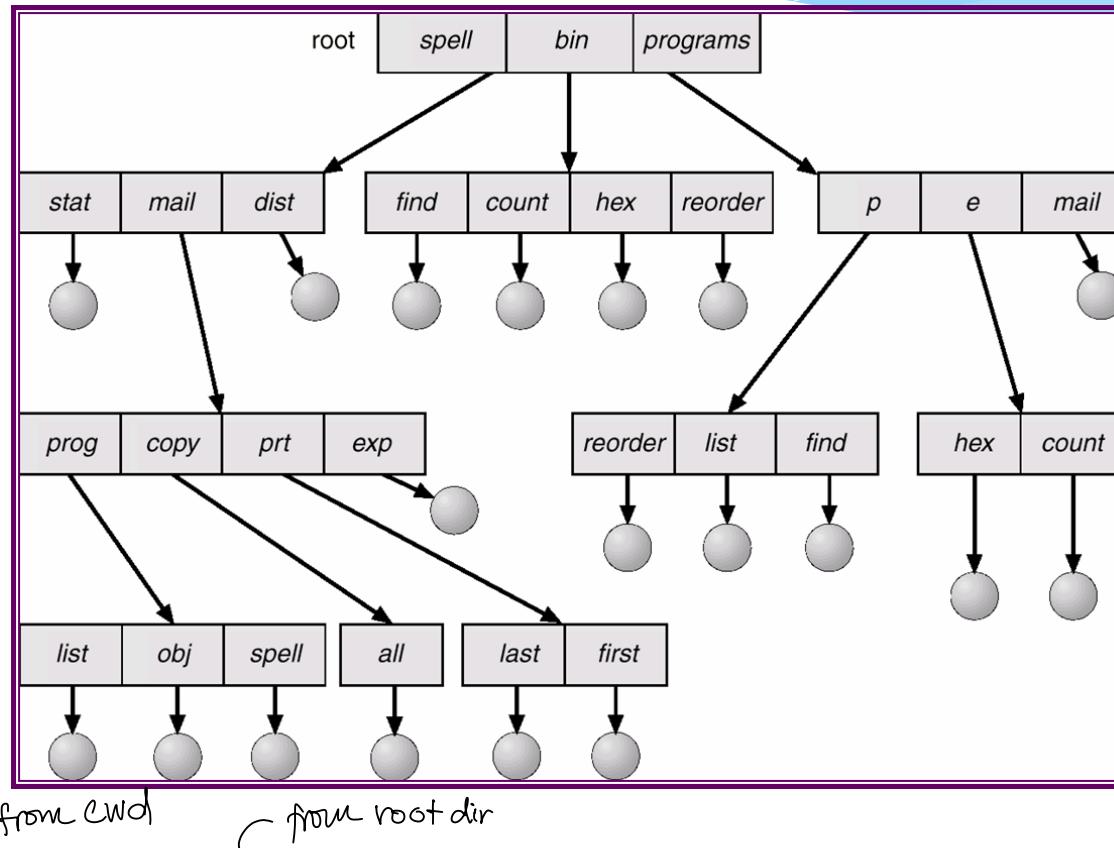
- Each user has his own private folder, when working, only work with private folder
- When log in, system check and allow user to work with private folder
- When a user is added
 - System create a new member in *Master file directory*
 - Create *User file directory*
- Solve the duplicate file name problem; Effect when user are independent
- Problem when user want to share a file

Chapter 4: File system management

1. File system

1.2. Directory structure

Tree structure directory



- A (relative/absolute) path to file exist
- Sub-directory is a file is treated specially (bit for marking)
- Operations create/delete/add... performed on current directory
 - Delete a sub-directory ⇒ delete its sub-tree

Chapter 4: File system management

1. File system

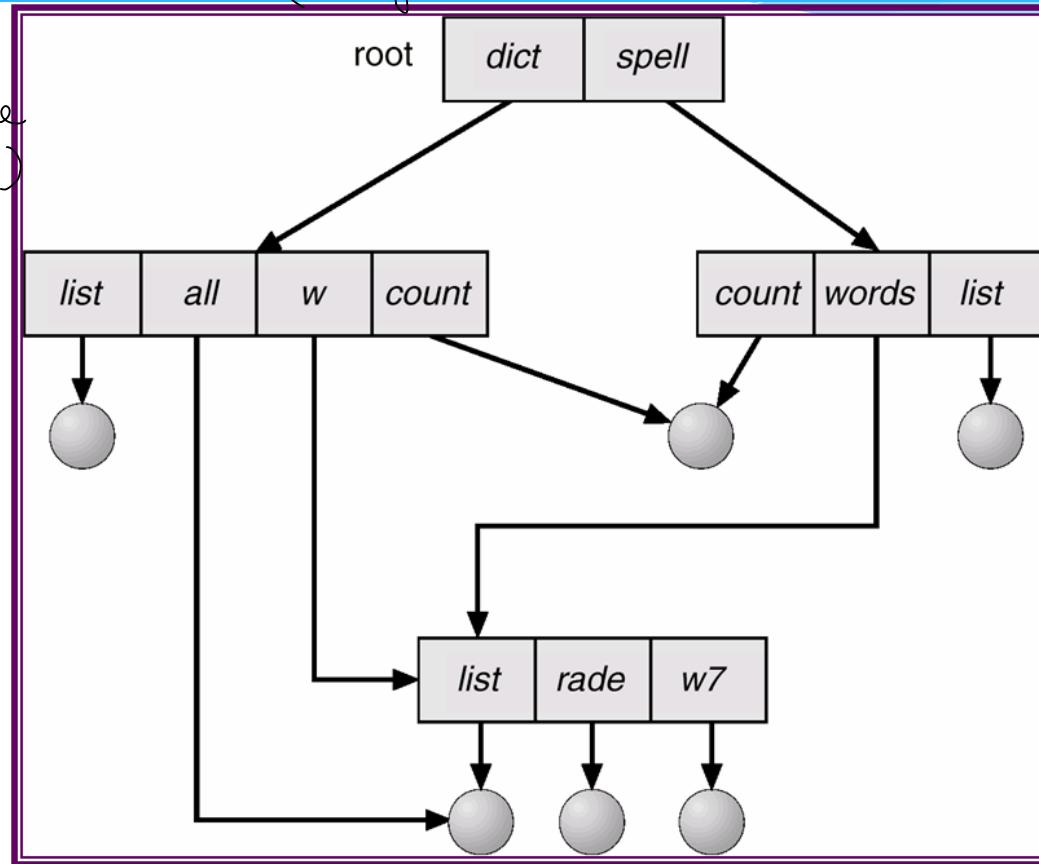
1.2. Directory structure

Sharing a directory (Acyclic - Graph Structure)

Nếu file đc share

→ Tạo con trỏ đến file
(đ/c vs mõi ng mõi)

⇒ Chỉ xoá khi
link cuối bị xoá



Check if file still had any links to it

- User can link to a file from other user
- When traversing the directory (backup), file can be traversed many time
- Delete file: link/ content (file-created-user /last link)

Chapter 4 File system management

- ① File system
- ② **File system's implementation**
- ③ Information organization on disk
- ④ FAT system

Chapter 4: File system management

2. File system's implementation

2.1 Directory implementation

- Directory implementation
- Storage area allocation methods
- Free storage area management

Chapter 4: File system management

2. File system's implementation

2.1 Directory implementation

Method

① Linear list with pointer to data blocks

- Simple for programming
- Time consuming when operate with directories
 - Must traverse all the list ⇐ Use binary tree?

file used as key for nodes

② Hash table – Hash table with linear list

- Reduce directory traversing time
- Require an effective hash function

$$h(\text{Name}) = \frac{\sum_{i=1}^{\text{Len}(\text{Name})} \text{ASCII}(\text{Name}[i])}{\text{Table_Size}}$$

i.e. filenames
'ABC'
'BAC'
'CBA'

- Collision problem: hash function return same result with 2 different file name
- Fixed table size problem: Increase size has to recalculate existed numbers (for all the files)

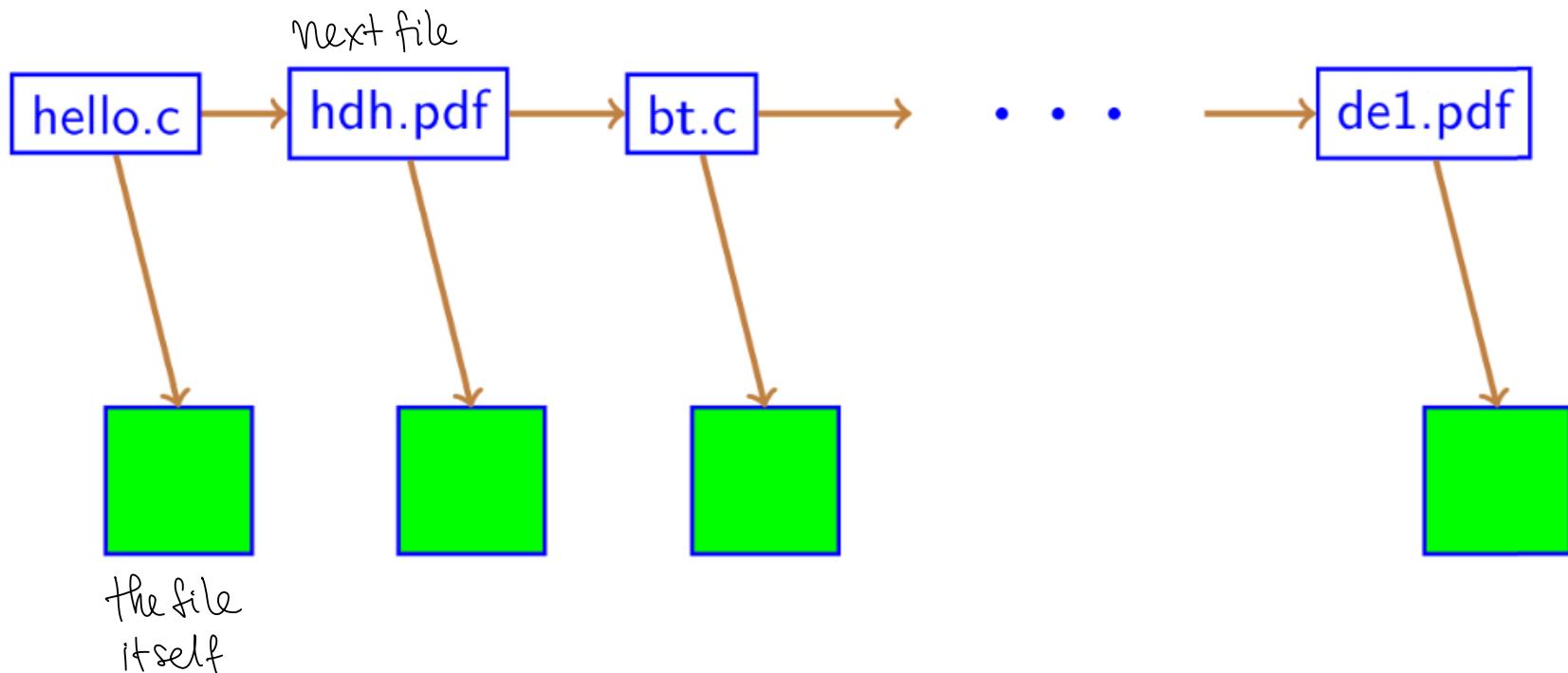
Chapter 4: File system management

2. File system's implementation

2.1 Directory implementation

Linear list

(multiple-link linked list)

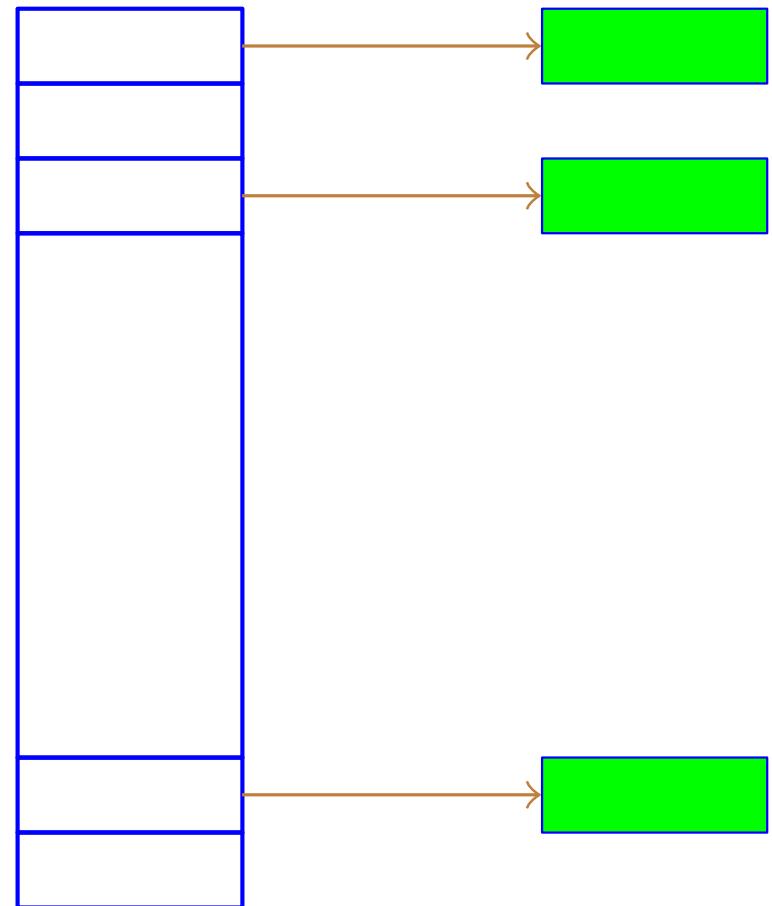


Chapter 4: File system management

2. File system's implementation

2.1 Directory implementation

Hash table



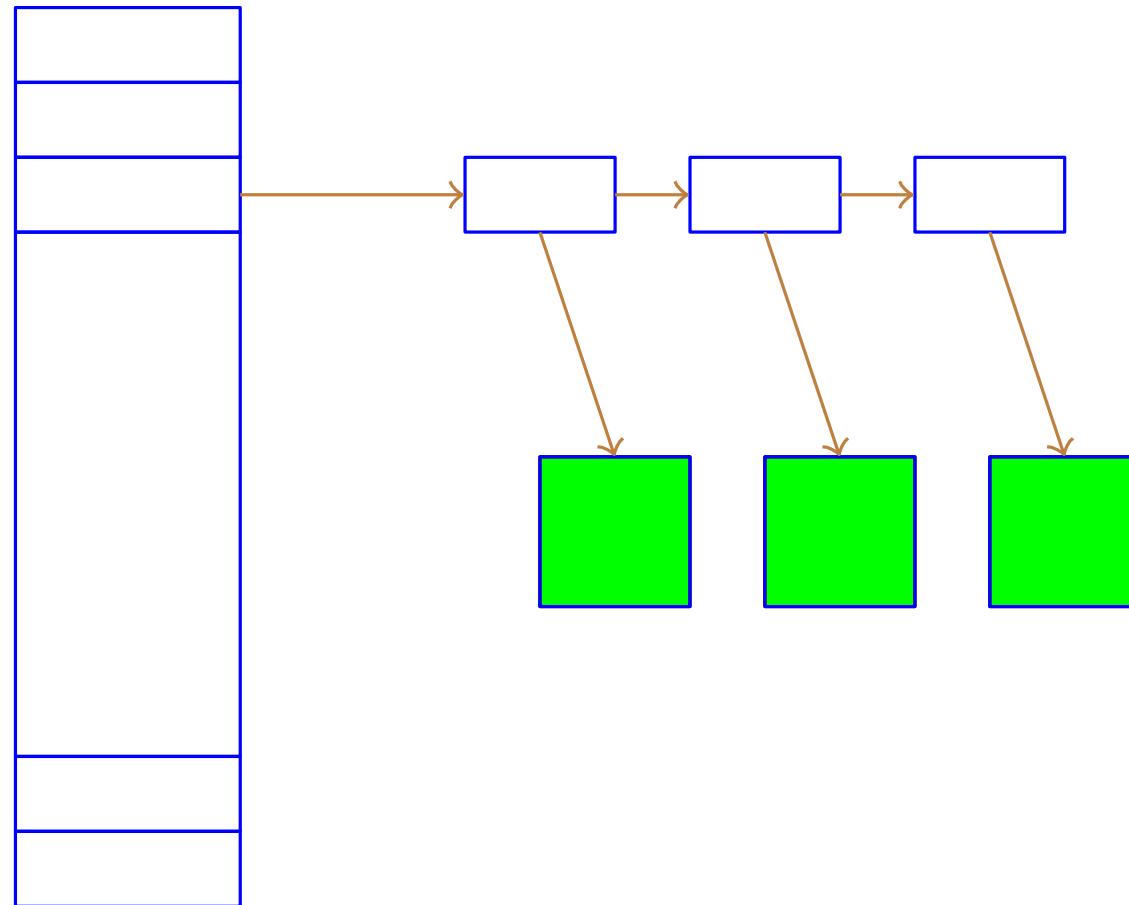
Chapter 4: File system management

2. File system's implementation

2.1 Directory implementation

Hash table

To solve the problem of potential collision!



Chapter 4: File system management

2. File system's implementation

2.2 Storage area allocation methods

- Directory implementation
- **Storage area allocation methods**
- Free storage area management

Chapter 4: File system management

2. File system's implementation

2.2 Storage area allocation methods

Methods

Objective

- Increase performance of sequence access
- Easy to randomly access file
- Easy to manage file

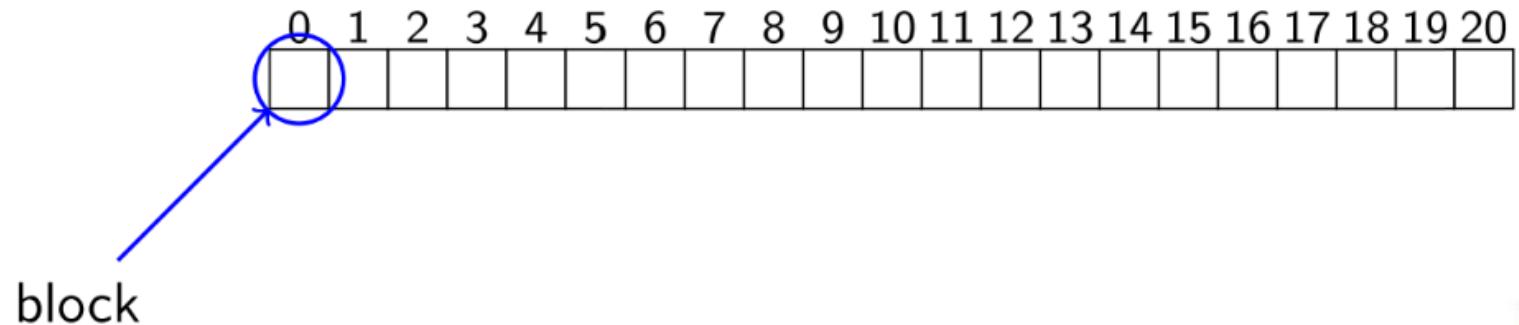
Methods

- Continuous Allocation
- Linked List Allocation
- Indexed Allocation

Continuous Allocation

Rule: File is allocated with continuous memory block

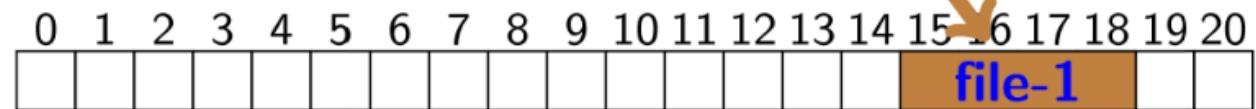
| File | <i>Pos</i> | <i>Size</i> |
|-----------|------------|-------------|
| file-1 | 15 | 4 |
| file-2 | 4 | 5 |
| file-3 | 11 | 3 |
| Directory | | |



Continuous Allocation

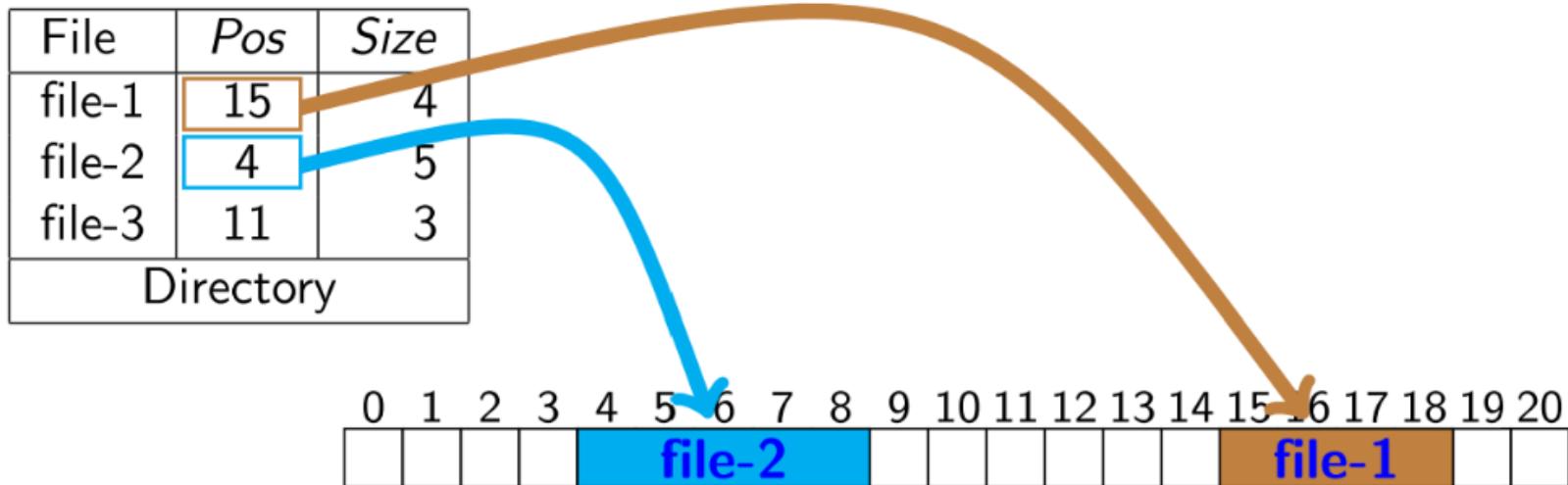
Rule: File is allocated with continuous memory block

| File | Pos | Size |
|-----------|-----|------|
| file-1 | 15 | 4 |
| file-2 | 4 | 5 |
| file-3 | 11 | 3 |
| Directory | | |



Continuous Allocation

Rule: File is allocated with continuous memory block



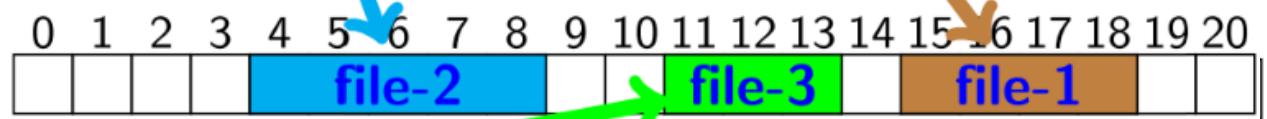
Continuous Allocation

Rule: File is allocated with continuous memory block

first memory block alloc to a file

| File | Pos | Size |
|--------|-----|------|
| file-1 | 15 | 4 |
| file-2 | 4 | 5 |
| file-3 | 11 | 3 |

Directory



Continuous Allocation

- File has the length n and begin at block b will occupy blocks $b, b + 1, \dots, b + n - 1$
 - Two block b and $b + 1$ are continuous
 - ⇒ No need to move the header (except the last sector)
 - ⇒ High speed access (fastest)
 - Allow directly access block i of file
 - ⇒ access block $b + i - 1$ on storage device
- Select empty space when there are request?
 - Strategies First-Fit /Worst Fit /Best Fit
 - External fragmentation phenomenon
- Difficult when the size of file increase

Chapter 4: File system management

2. File system's implementation

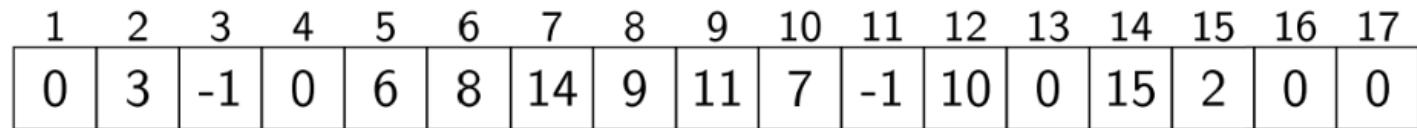
2.2 Storage area allocation methods

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block

| File | <i>Pos</i> | <i>End</i> |
|-----------|------------|------------|
| abc | 12 | 3 |
| def | 5 | 11 |
| Directory | | |



Chapter 4: File system management

2. File system's implementation

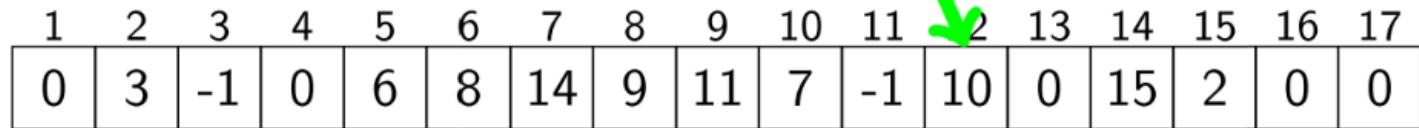
2.2 Storage area allocation methods

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block

| File | Pos | End |
|-----------|-----|-----|
| abc | 12 | 3 |
| def | 5 | 11 |
| Directory | | |



Chapter 4: File system management

2. File system's implementation

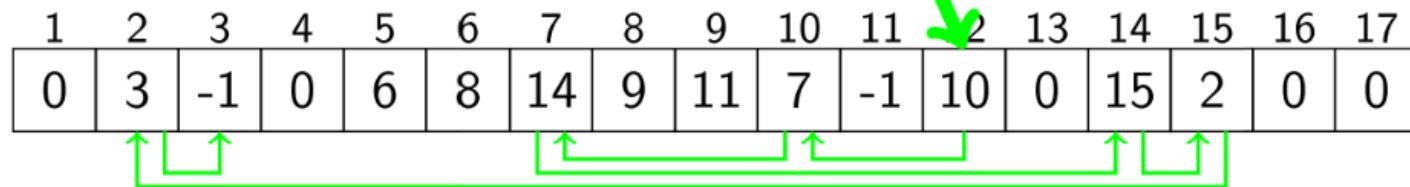
2.2 Storage area allocation methods

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block

| File | Pos | End |
|-----------|-----|-----|
| abc | 12 | 3 |
| def | 5 | 11 |
| Directory | | |



Chapter 4: File system management

2. File system's implementation

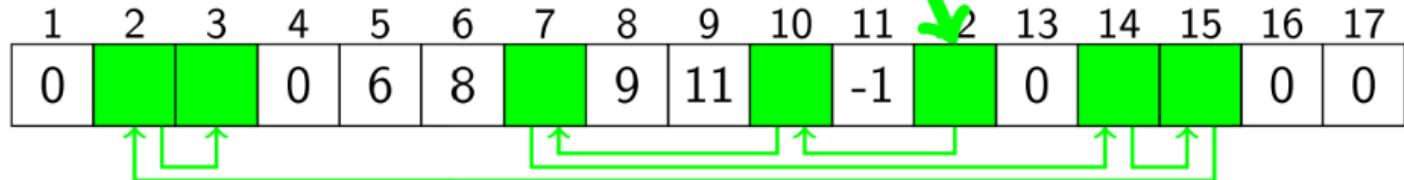
2.2 Storage area allocation methods

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block

| File | Pos | End |
|-----------|-----|-----|
| abc | 12 | 3 |
| def | 5 | 11 |
| Directory | | |

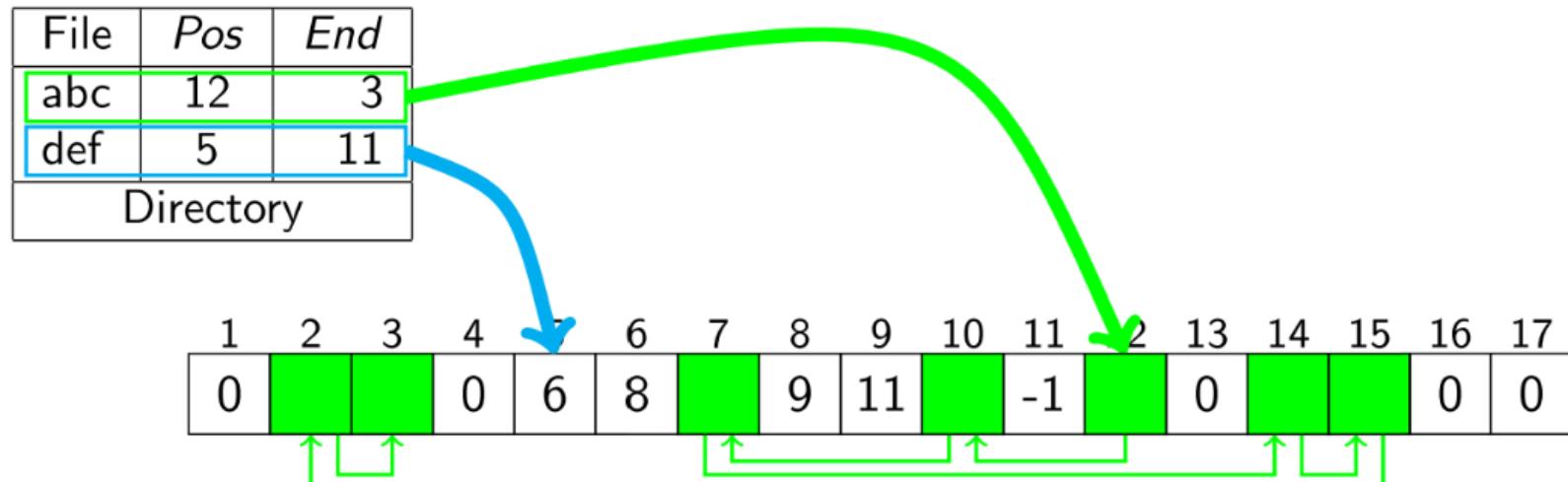


Files abc consists of 7 blocks: 12, 10, 7, 14, 15, 2, 3

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block



Files abc consists of 7 blocks: 12, 10, 7, 14, 15, 2, 3

Chapter 4: File system management

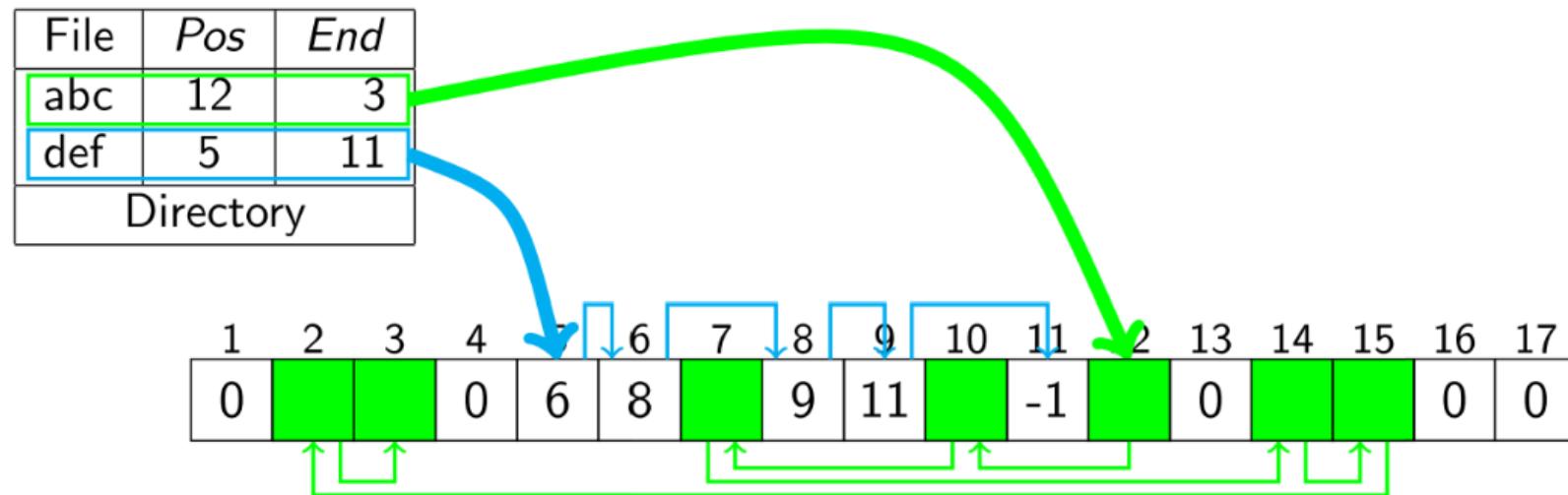
2. File system's implementation

2.2 Storage area allocation methods

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block



Files abc consists of 7 blocks: 12, 10, 7, 14, 15, 2, 3

Chapter 4: File system management

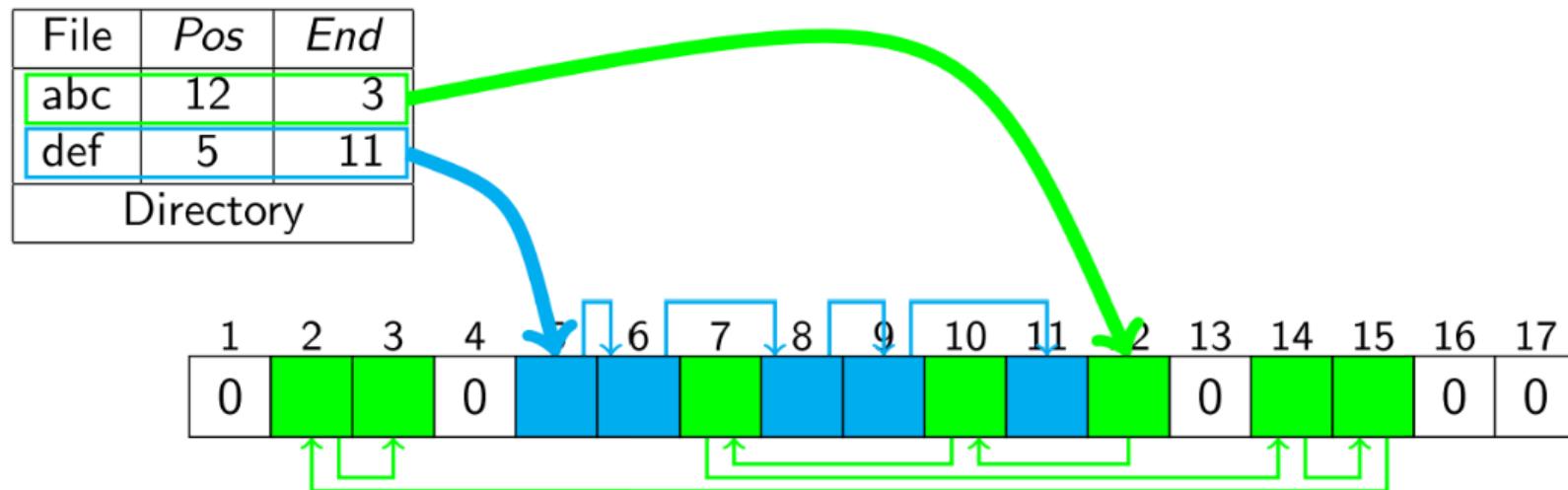
2. File system's implementation

2.2 Storage area allocation methods

Linked List Allocation

Rule: File is allocated with non continuous memory blocks.

End of each block is a pointer, point to next block



Files abc consists of 7 blocks: 12, 10, 7, 14, 15, 2, 3

Files def consists of 5 blocks: 5, 6, 8, 9, 11

Linked List Allocation

- Only effective for sequent access file
- To access block n , must traverse $n - 1$ blocks before it
 - Blocks are noncontiguous, has to relocation from start
 - Slowly access speed
- Blocks in file are linked by pointers -> if pointer broke?
 - Lost data due to link to block is lost
 - Link to block without data or block belongs to other file
- **Solution:** Apply many pointer in each block \Rightarrow Consume memory
- Apply: FAT
 - Utilized as a linked list
 - Combined of many members, each member corresponding to a block
 - Each member in FAT, contain next block of file
 - Last block has special value (FFFF)
 - Error block has value (FFF7)
 - Unused block has value (0)
 - **Location** field in **file record** contains the first block of file

Chapter 4: File system management

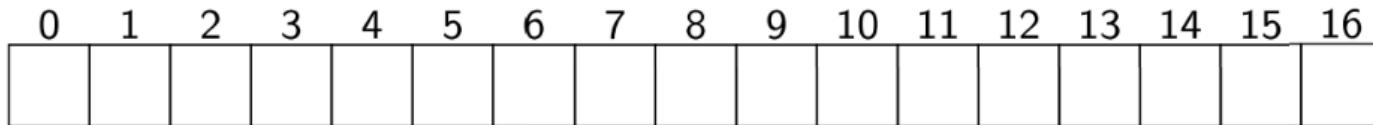
2. File system's implementation

2.2 Storage area allocation methods

Index allocation

Rule: Each file has a main index block which contains list of file block

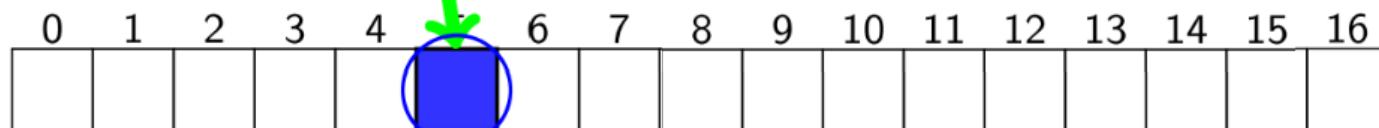
| File | <i>Index block</i> |
|-----------|--------------------|
| abc | 5 |
| def | 12 |
| Directory | |



Index allocation

Rule: Each file has a main index block which contains list of file block

| File | <i>Index block</i> |
|-----------|--------------------|
| abc | 5 |
| def | 12 |
| Directory | |

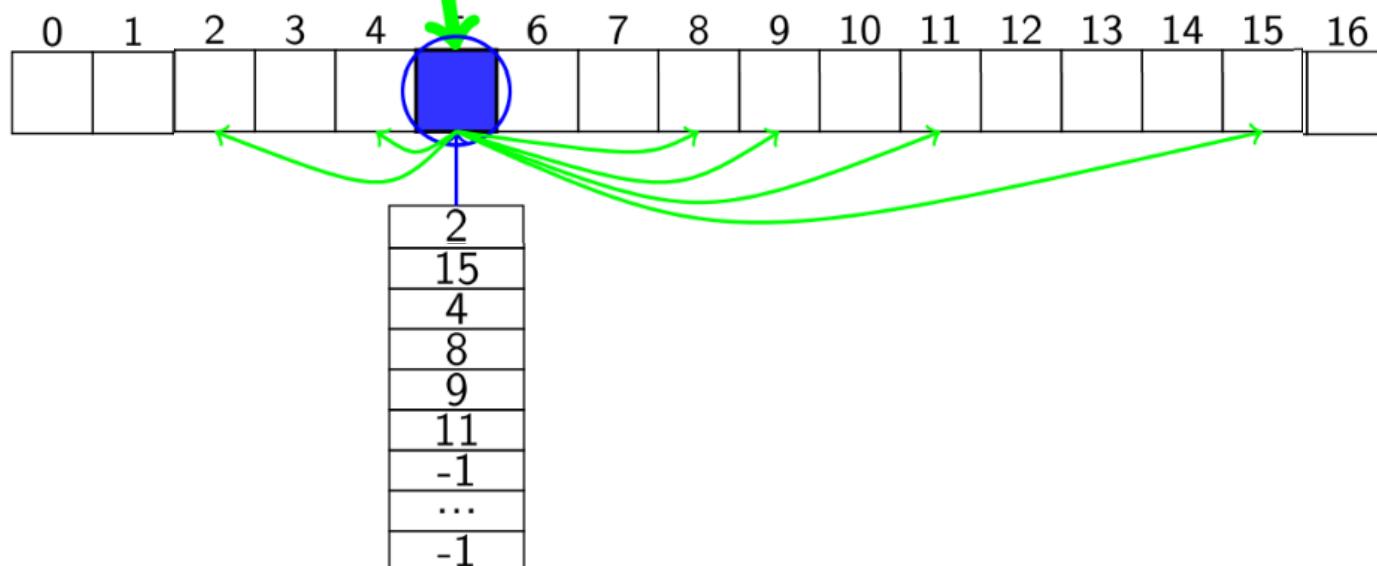


| |
|-----|
| 2 |
| 15 |
| 4 |
| 8 |
| 9 |
| 11 |
| -1 |
| ... |
| -1 |

Index allocation

Rule: Each file has a main index block which contains list of file block

| File | <i>Index block</i> |
|-----------|--------------------|
| abc | 5 |
| def | 12 |
| Directory | |



Index allocation

Rule: Each file has a main index block which contains list of file block

| File | <i>Index block</i> |
|-----------|--------------------|
| abc | 5 |
| def | 12 |
| Directory | |



File abc consists of 6 blocks: 2, 15, 4, 8, 9, 11

Index allocation

Rule: Each file has a main index block which contains list of file block

| File | <i>Index block</i> |
|-----------|--------------------|
| abc | 5 |
| def | 12 |
| Directory | |



Files abc consists of 6 blocks: 2, 15, 4, 8, 9, 11

| |
|-----|
| 10 |
| 13 |
| -1 |
| ... |
| -1 |

Chapter 4: File system management

2. File system's implementation

2.2 Storage area allocation methods

Index allocation

Rule: Each file has a main index block which contains list of file block

| File | <i>Index block</i> |
|-----------|--------------------|
| abc | 5 |
| def | 12 |
| Directory | |



File abc consists of 6 blocks: 2, 15, 4, 8, 9, 11

File def consists of 2 blocks: 10, 13

Index allocation

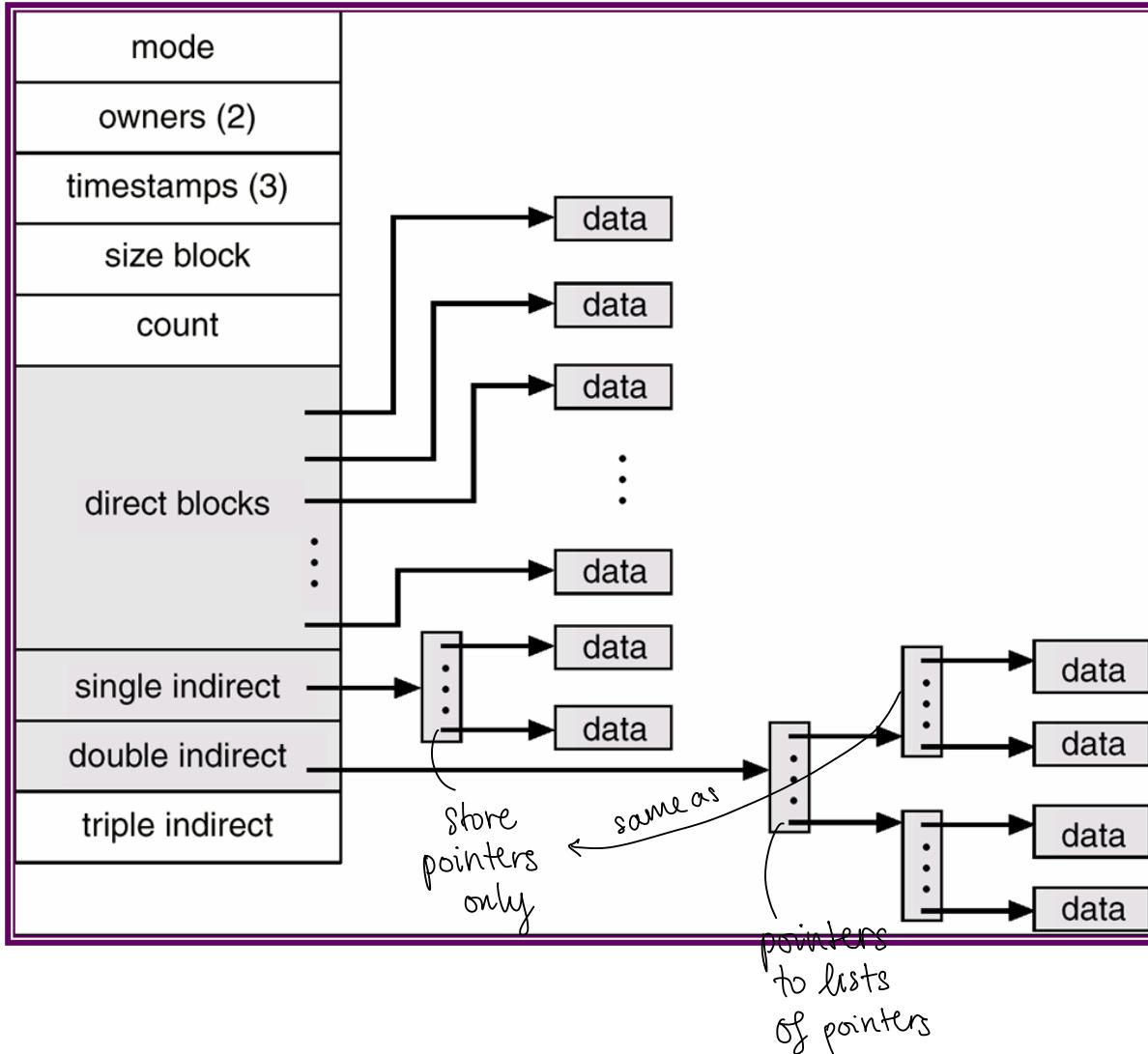
- Member **i** of index block point to block **i** of file
 - Read block **i** use pointer declared at member **i** of index block
 - Create file, members of index block has null value (-1)
 - Require block **i**, block address is allocated, putted into member **i**
 - Remark
 - No external fragmentation
 - Allow direct access
 - Require index block: file has small size require 2 blocks
 - Block for data
 - Block for index (use 1 member)
- Large
size file
problem ?*
- ↓
- Linked map
 - Connects index block
 - Last member of index block point to other index block if it's necessary
 - Multi level Index
 - Use an index block point to other index block

Chapter 4: File system management

2. File system's implementation

2.2 Storage area allocation methods

Link map (UNIX)



- 12 direct block point to data block
- Single indirect block contains address of direct block
- Double indirect block contains address of Single indirect block
- Triple indirect block contain address of Double indirect

Chapter 4: File system management

2. File system's implementation

2.3 Free storage area management

- Directory implementation
- Storage area allocation methods
- **Free storage area management**

Chapter 4: File system management

2. File system's implementation

2. 3 Free storage area management

Methods

- Bit vector
 - Each block represented by 1 bit (1: free; 0: allocated)
 - Easy to find n contiguous memory block
 - Need instruction to work with bit
- Link list
 - Hold pointer to first free disk block
 - This memory block contain pointer to next free disk block
 - Not effective when traversing the list
- Grouping
 - Hold address of n free block in the first free block
 - n – 1 first free block, block n contain address of next n free block
 - Pros: Fast to find free memory area
- Counting
 - Due to memory block continuously allocated and free concurrently
 - Rule: Store address of the first free block and size of contiguous memory area in free-memory-area management list
 - Effective when the counter larger than 1

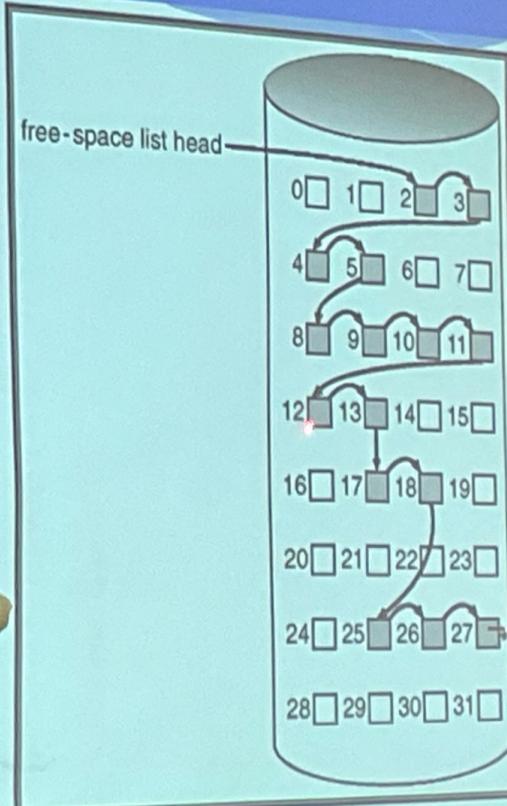


2. File system management
2. File system's implementation
2. 3 Free storage area management

Methods

- Link list

- Hold pointer to first free disk block
- This memory block contain pointer to next free disk block effect when user request



Chapter 4 File system management

- ① File system
- ② File system's implementation
- ③ Information organization on disk
- ④ FAT system

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

- Disk's physical structure
- Disk's logical structure

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Floppy disk $5\frac{1}{4}$

A: disk driver

used to read

this type →
of disk

B also ↑

C: where Windows
started



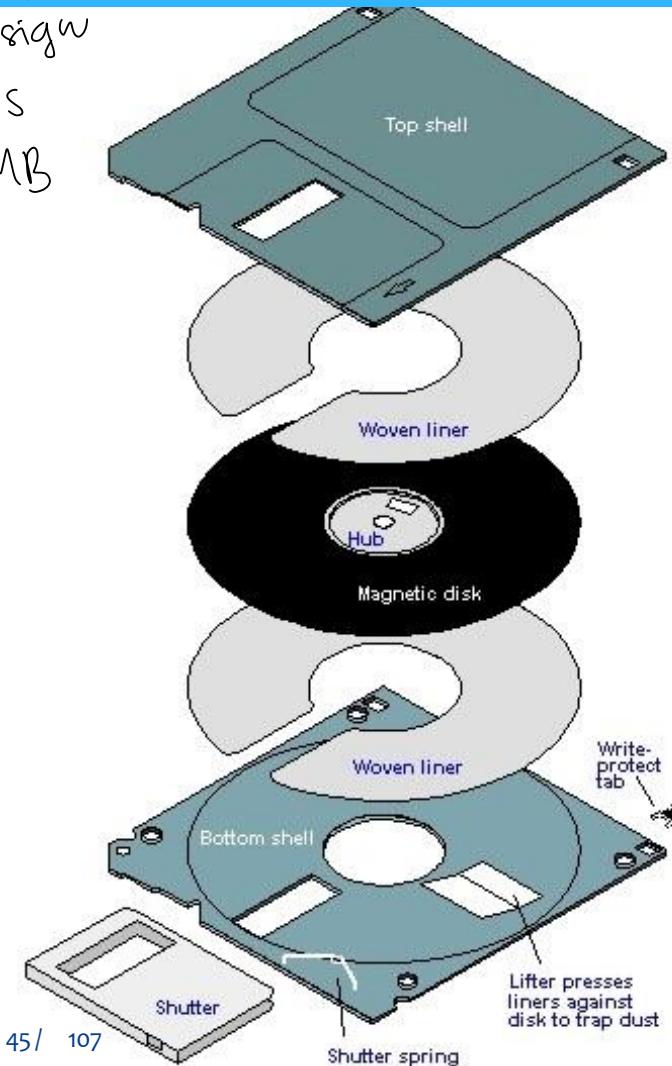
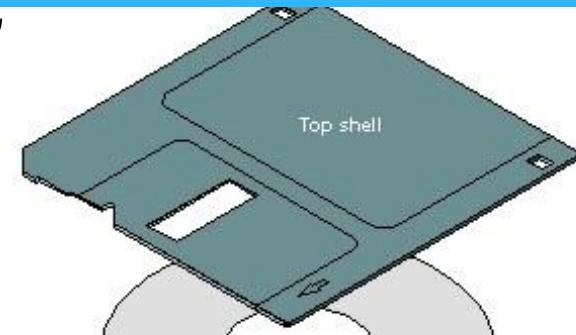
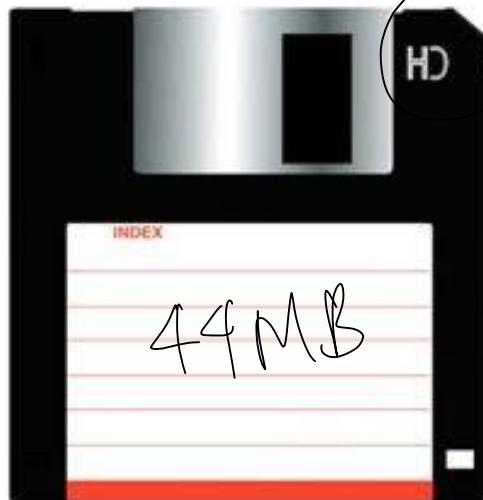
Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

still used in Japan &
airplane controlling sys

Floppy disk $3\frac{1}{2}$

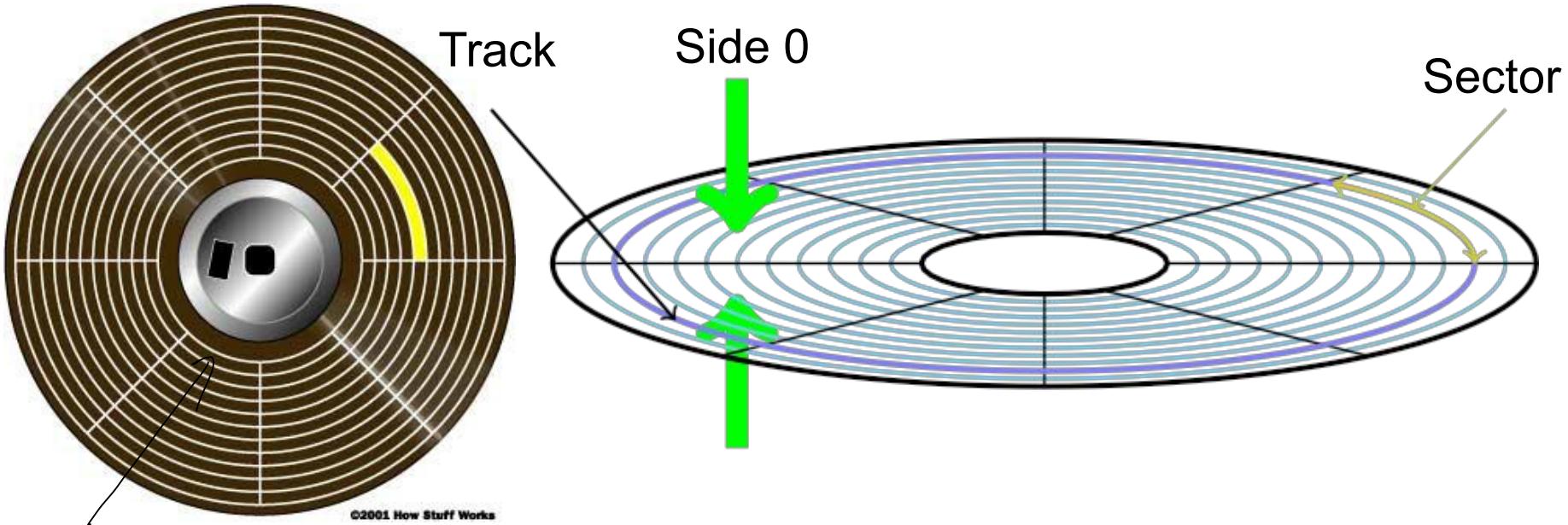


Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Floppy disk's physical structure

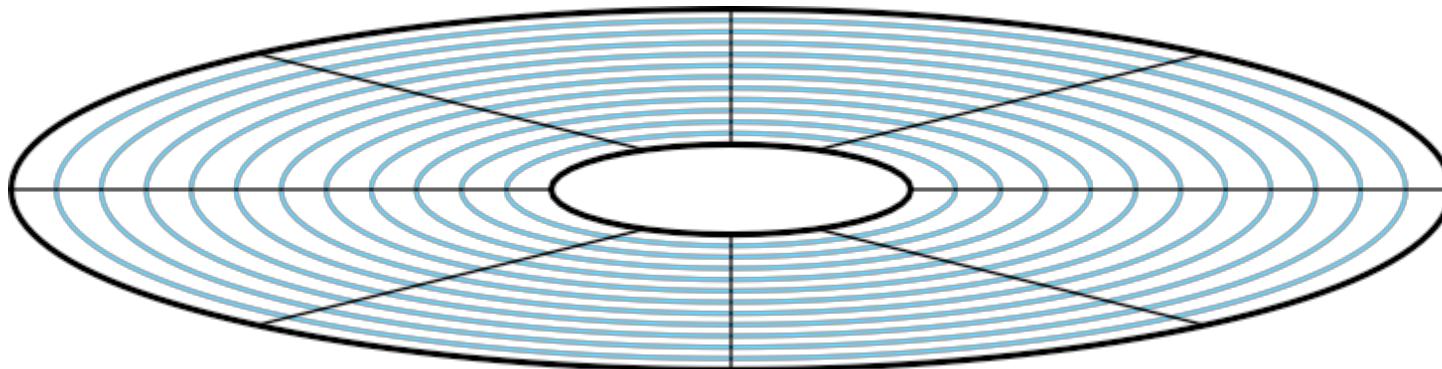


- Track: concentric circles around the disk
 - Numbered 0, 1, . . . From outer to inner
- Side. Each side is read by a Header
 - Headers are numbered 0, 1
- Sector
 - Numbered 1, 2, . . .

Information positioning on floppy disk

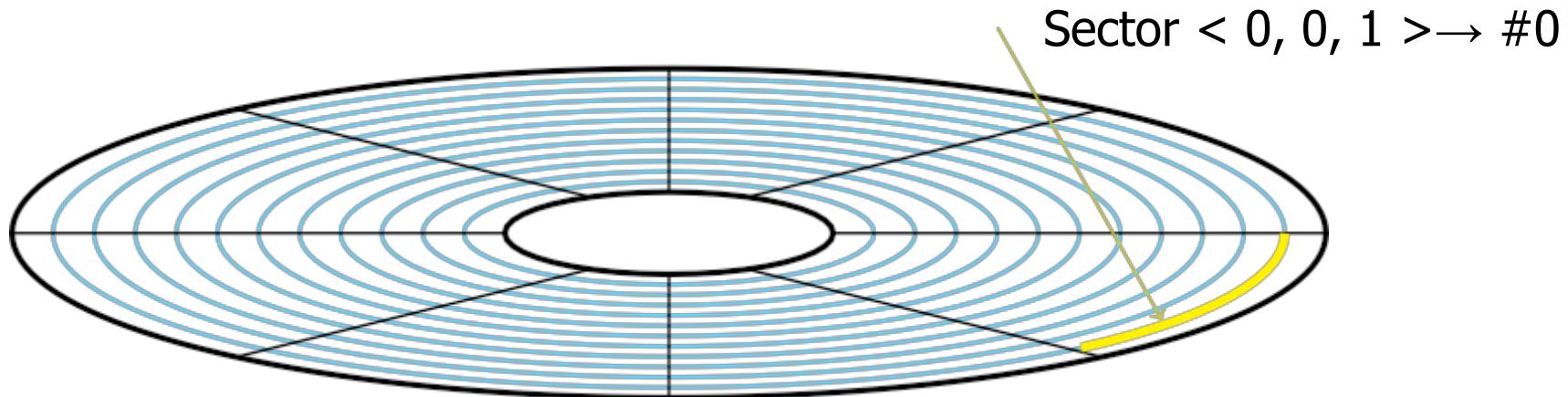
- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector

conversion?



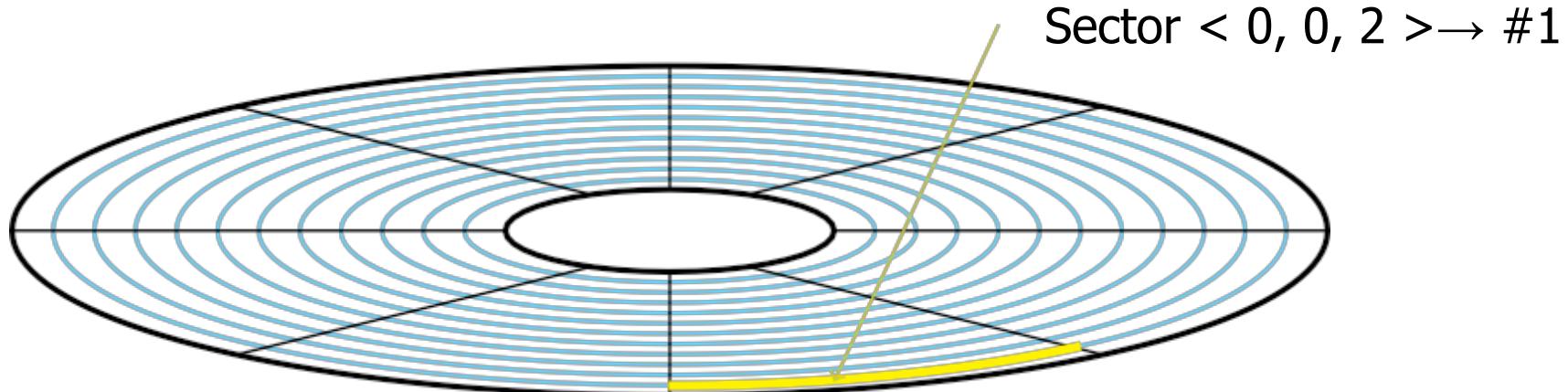
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



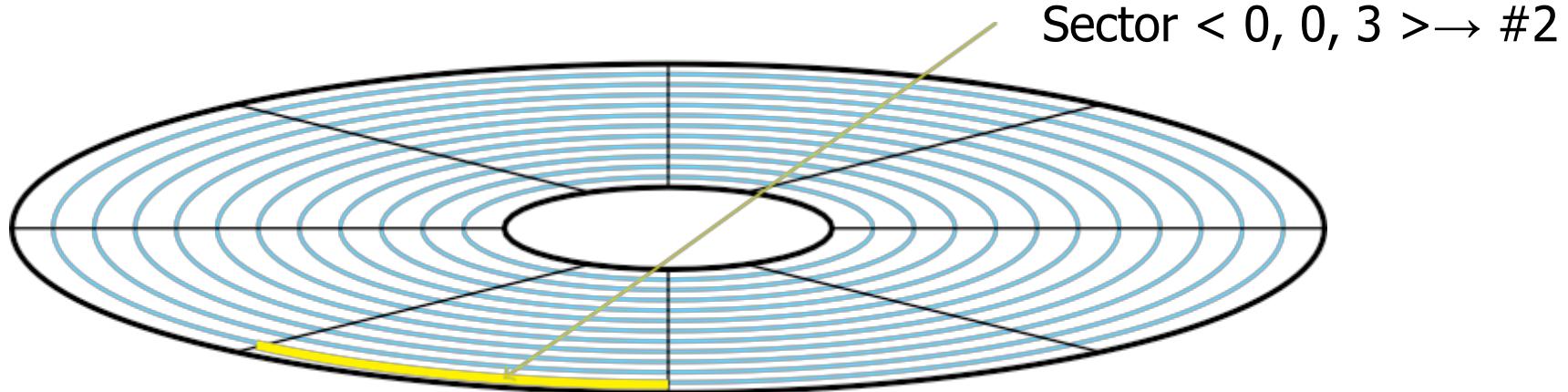
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



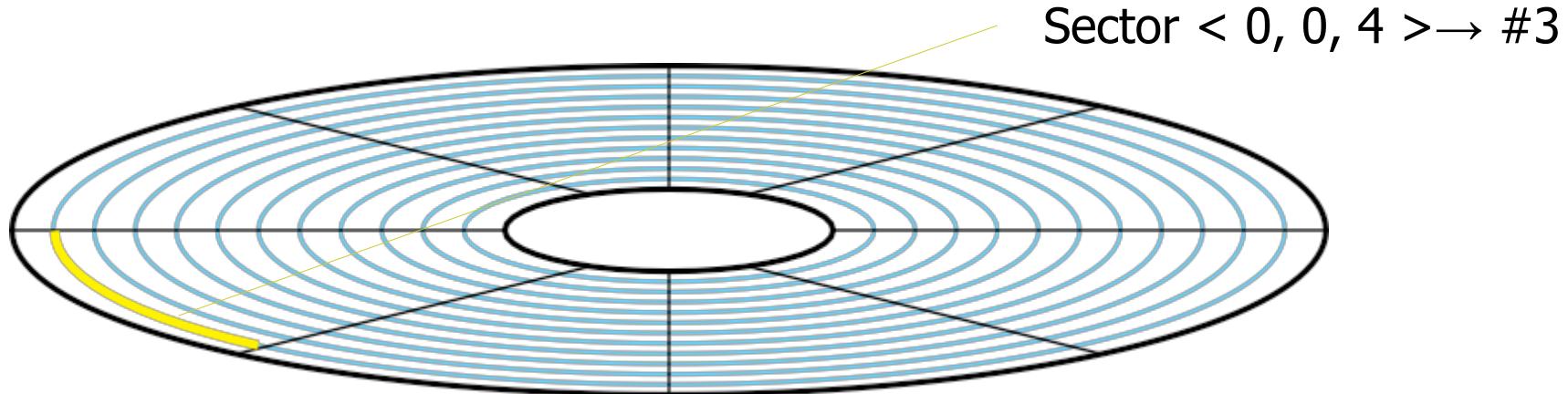
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



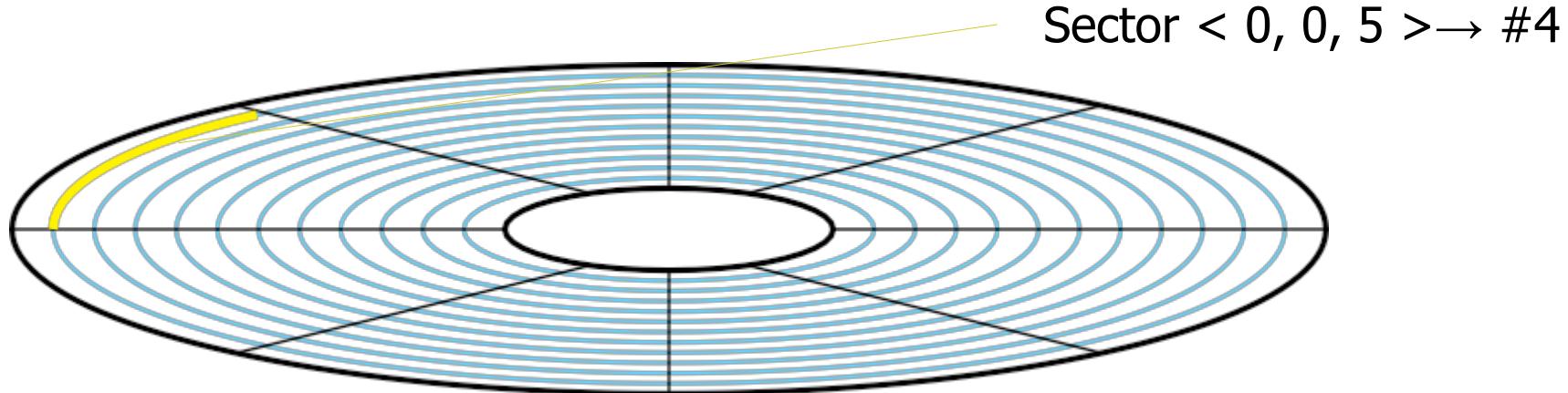
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



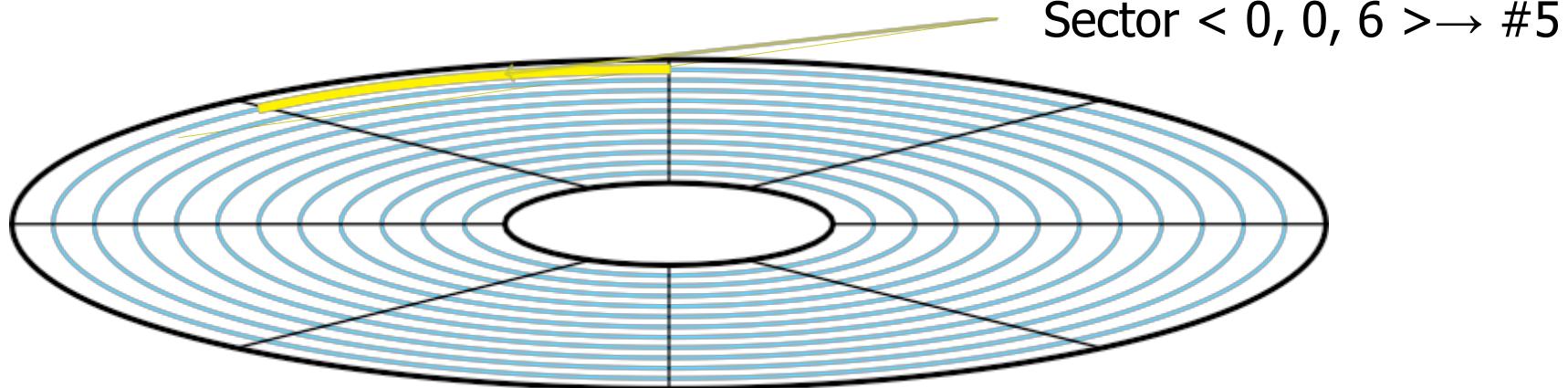
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



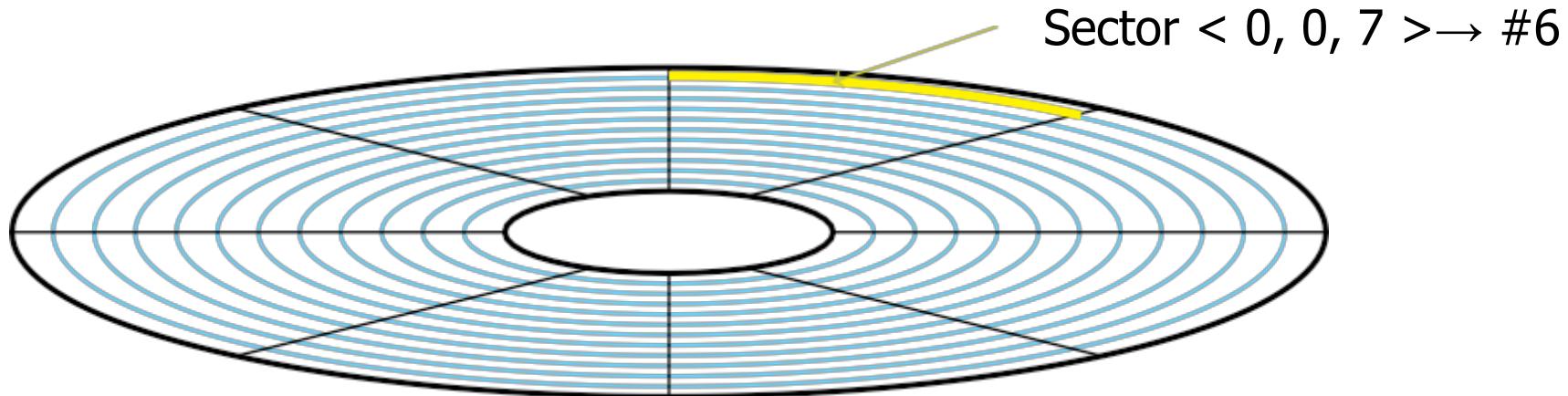
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



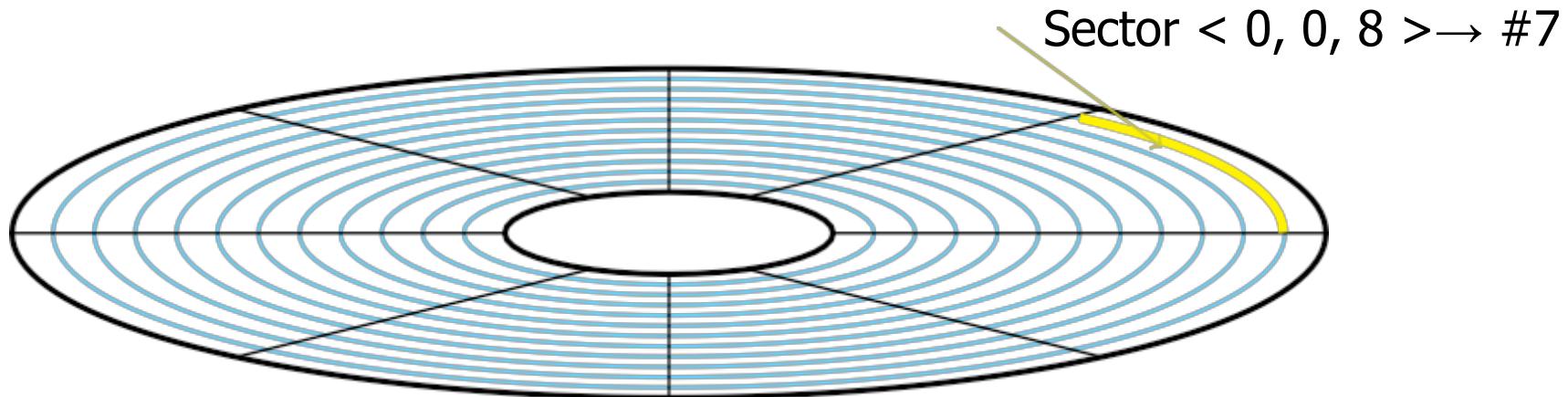
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



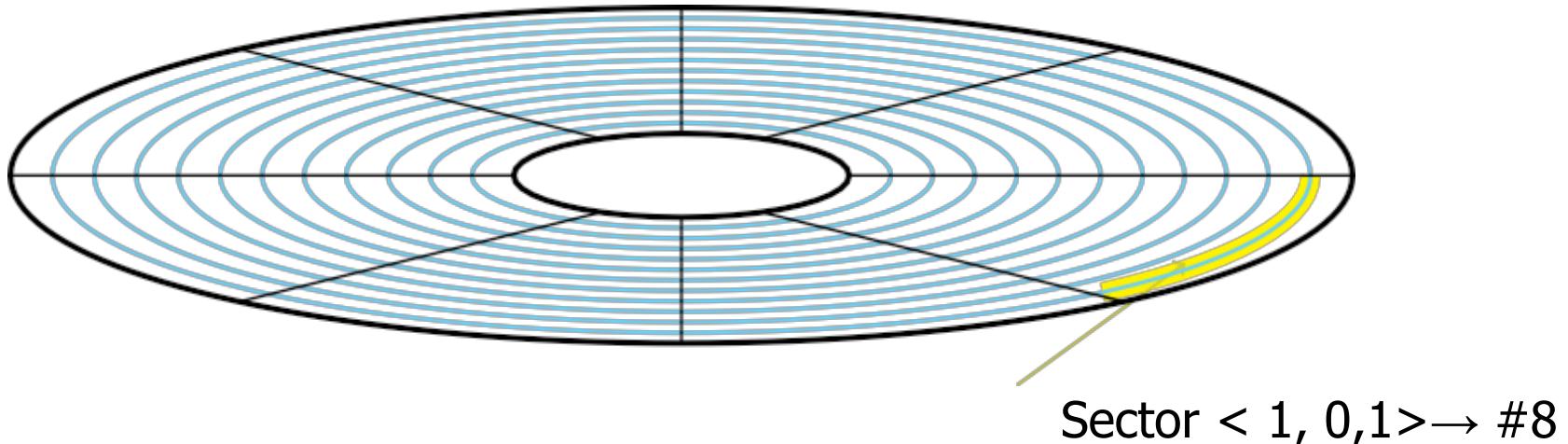
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



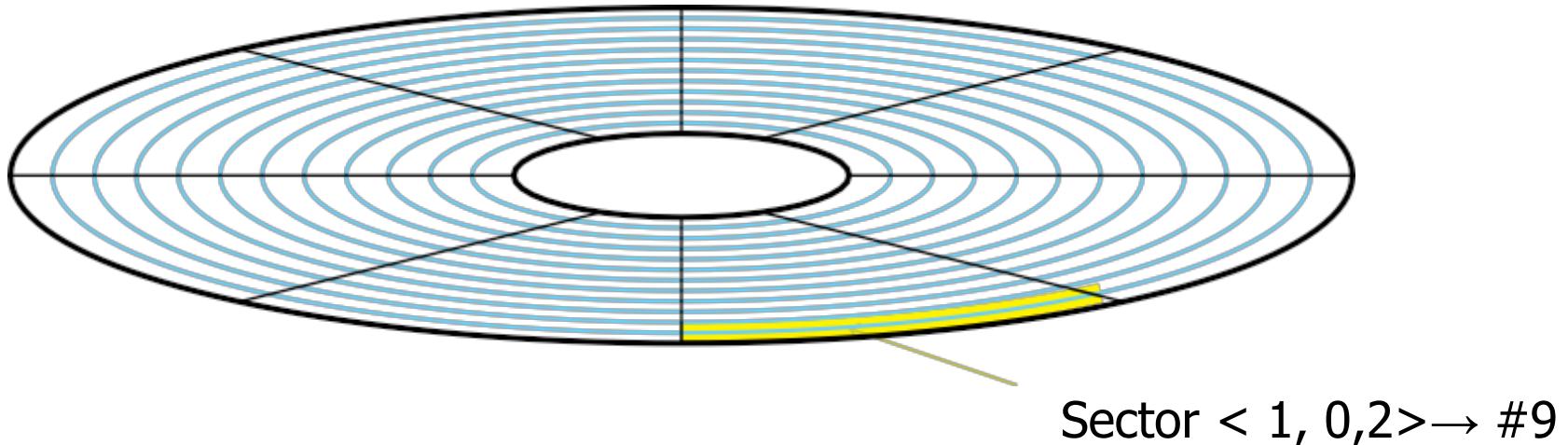
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



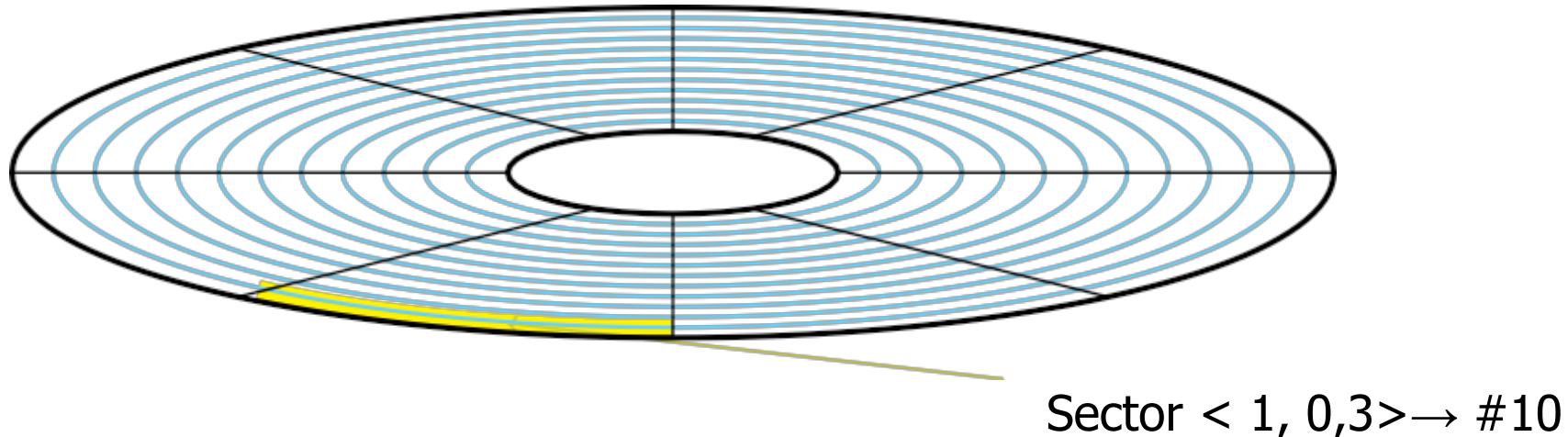
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



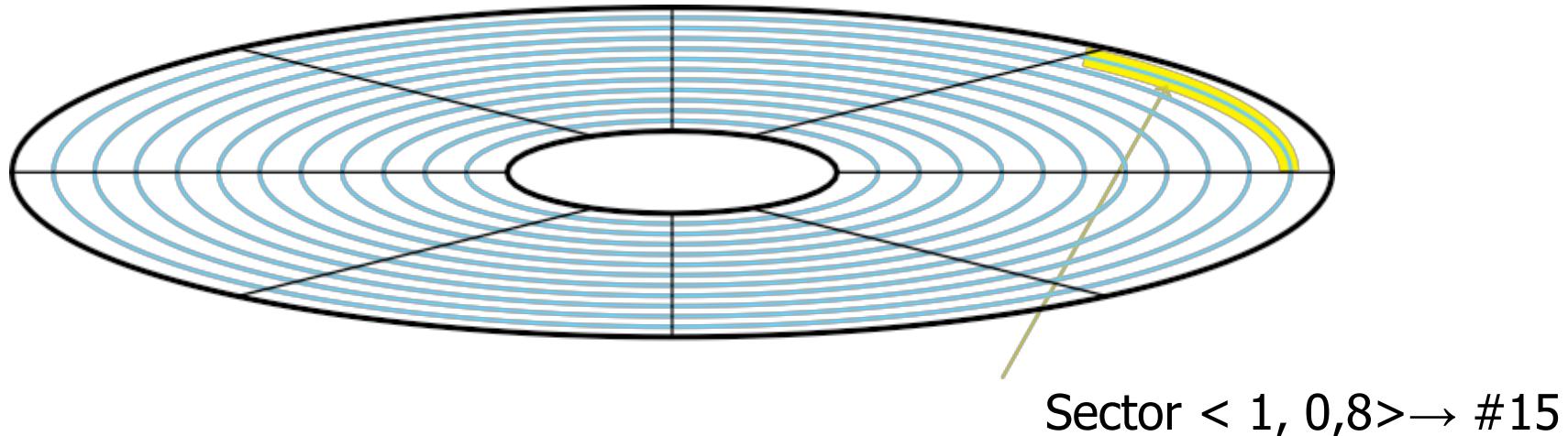
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



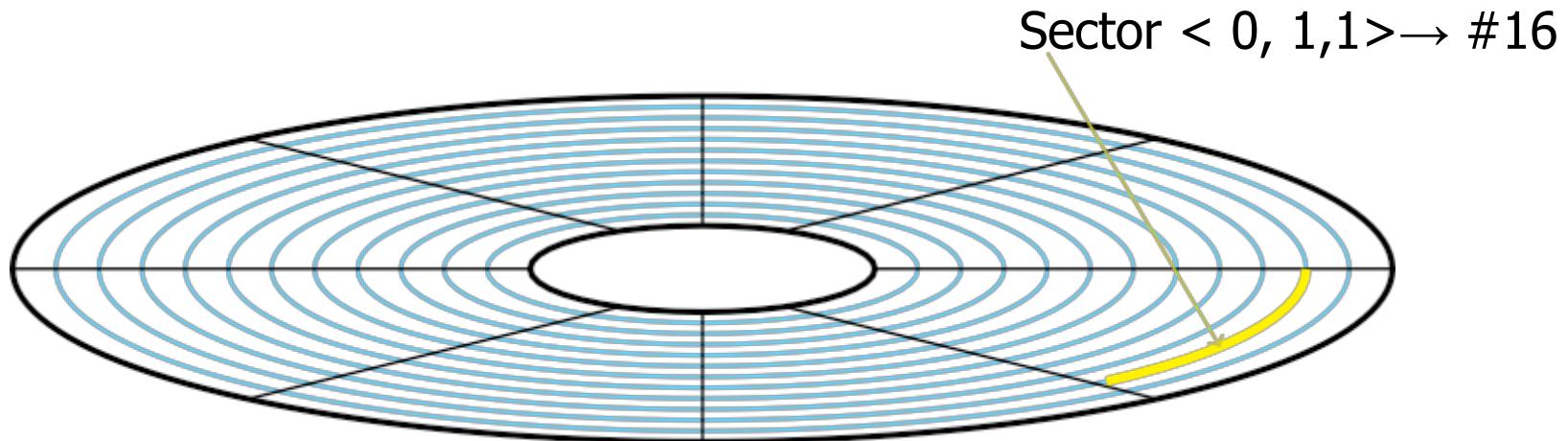
Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



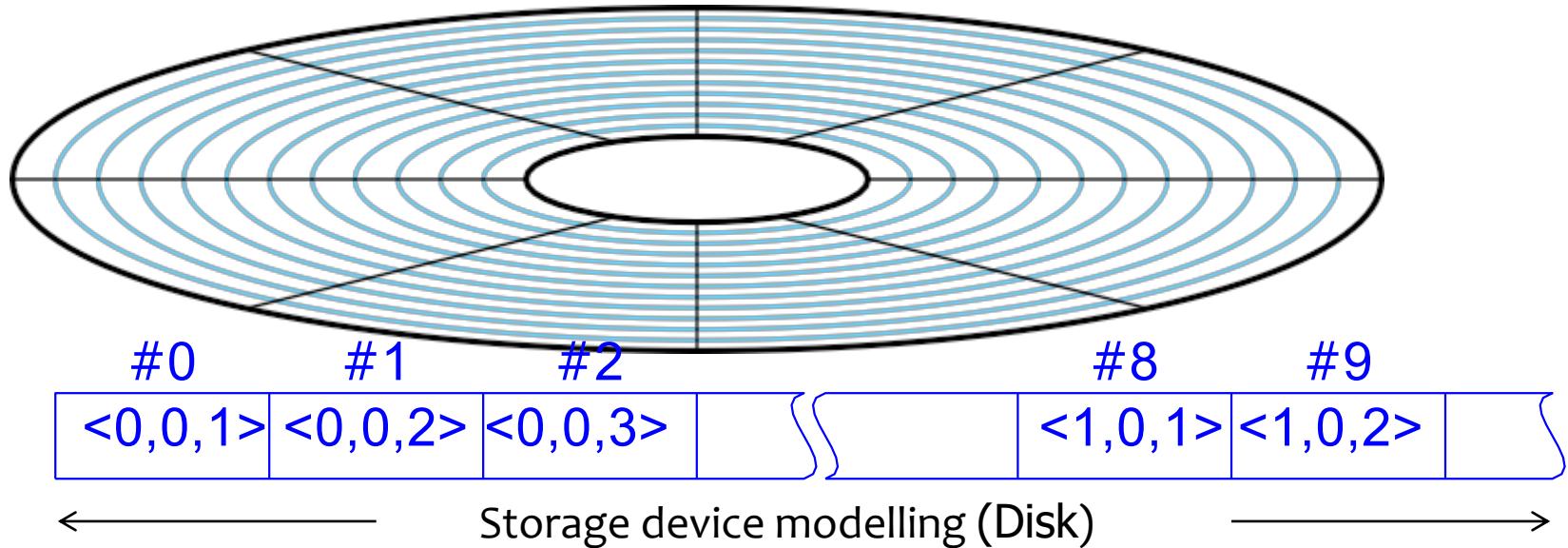
Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Information positioning on floppy disk

- Sector: information unit for working with disk
- Sector defined via 3 dimension position: Header, Track, Sector
 - Example: Floppy disk's Boot Sector: Sector $<0, 0, 1>$
- Sector defined via sector number (1 direction position)
 - Relative position from disk's first sector



Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Hard disk



Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Hard disk

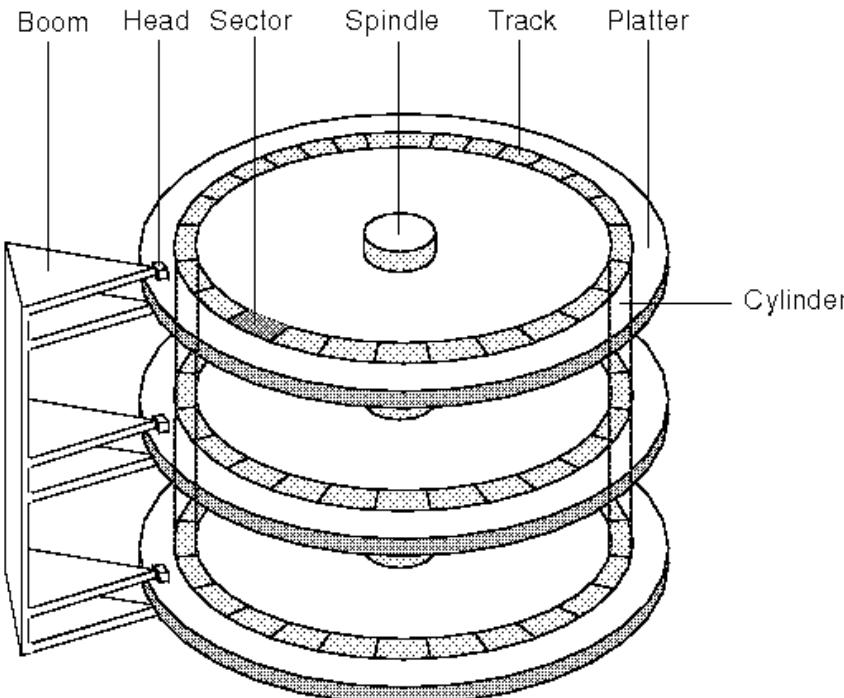


Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Hard disk's physical structure



Structure

- Consist of many headers, numbered from 0,1
- Tracks with the same radius create a cylinder, numbered from 0, 1, ...
- Sectors on each side of cylinder, numbered from 1,2,...

(as in floppy) track w/
the same
radius

Information positioning

- 3 dimensions (H, C, S)
- 1 dimension: sector's number
 - Rule similar to floppy disk: Sector → Header → Cylinder

Accessing sector on disk

13th

- Sector: information unit to work with disks
- Can access (read/write/format/...) to each sector
- Access via BIOS 13h interrupt (function 2, 3, 5,...)
 - Not depend on operating system
 - Sector is determined by address <H,C,S>
- Access via system call
 - Operating system's interrupt
 - Example: MSDOS provide 25h/26h interrupt allow read/write sectors at linear address
 - Use WIN32 API functions
 - CreateFile()/ReadFile()/WriteFile()...

th th

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Interrupt 13h

| Register | Meaning |
|-----------|--|
| AH | 2h:Read sector; 3h: write Sector |
| AL | Number of sector to read Sectors must be on the same side, track |
| DH | Header number |
| DL | Disk's number. 0h:A; 80h: First hard disk; 81h Second hard disk 2 |
| CH | Track/Cylinder's number (Use 10 bits, borrow 2 high bits from CL) |
| CL | Sector's number (only use 6 lower bits) |
| ES:BX | Point to buffer area where data will be read (when AH=2h) or write to disk (when AH=3h) |
| CarryFlag | CF=0 no error; CL contains number of sector read CF=1 Had error, AH contains error's code |

WinXP limit the use on interrupt 13h to direct access

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Use interrupt 13h (Example)

```
#include <stdio.h>
#include <dos.h>
int main(int argc, char *argv[]){
    union REGS regs;
    struct SREGS sregs;
    int Buf[512];
    int i;
    regs.h.ah = 0x02;
    regs.h.al = 0x01;
    regs.h.dh = 0x00;
    regs.h.dl = 0x80;
    regs.h.ch = 0x00;
    regs.h.cl = 0x01;
    regs.x.bx = FP_OFF(Buf);
    sregs.es = FP_SEG(Buf);
    int86x(0x13,&regs,&regs,&sregs);
    for(i=0;i<512;i++) printf("%4X",Buf[i]);
    return 0;
}
```

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Use WIN32 API

- **HANDLE CreateFile(. . .)**: Open file/IO device
 - LPCTSTR lpFileName, ⇒ file/IO device's name
 - " " ? : " Partition / Drive C
 - " PhysicalDrive0" First hard drive
 - DWORD dwDesiredAccess, ⇒ Operation with device
 - DWORD dwShareMode, ⇒ Allow sharing
 - LPSECURITY_ATTRIBUTES lpSecurityAttributes (NULL),
 - DWORD dwCreationDisposition, ⇒ Operation to perform
 - DWORD dwFlagsAndAttributes, ⇒ Attribute
 - HANDLE hTemplateFile (NULL)
- **BOOL ReadFile(. . .)**
 - HANDLE hFile, ⇒ File to read
 - LPVOID lpBuffer, ⇒ Buffer to store data
 - DWORD nNumberOfBytesToRead, ⇒ number of byte to read
 - LPDWORD lpNumberOfBytesRead, ⇒ number of read bytes
 - LPOVERLAPPED lpOverlapped (NULL)
- **BOOL WriteFile(. . .)** ⇒ Parameters similar to ReadFile()

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Use WIN32 API (Example)

```
#include <windows.h>
#include <stdio.h>
int main(int argc, char *argv[]){
    HANDLE hDisk;
    BYTE Buf[512];
    int byteread,i;
    hDisk=CreateFile("¥¥¥¥.¥¥PhysicalDrive0",GENERIC_READ,
                    FILE_SHARE_READ | FILE_SHARE_WRITE,
                    NULL, OPEN_EXISTING,0,NULL);
    if (hDisk==INVALID_HANDLE_VALUE) printf("Device's error");
    else {
        ReadFile(hDisk,Buf,512,&byteread,NULL);
        for(i=0;i<512;i++) printf("%4X",Buf[i]);
        CloseHandle(hDisk);
    }
    return 0;
}
```

Chapter 4: File system management

3. Information organization on disk

3.1 Disk's physical structure

Result

(Exam) Decode this information?

Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

- Disk's physical structure
- Disk's logical structure

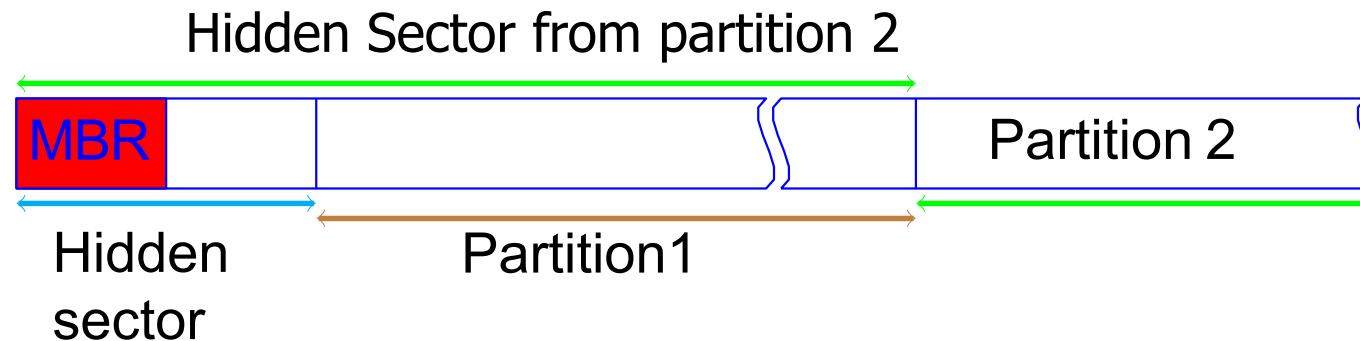
Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

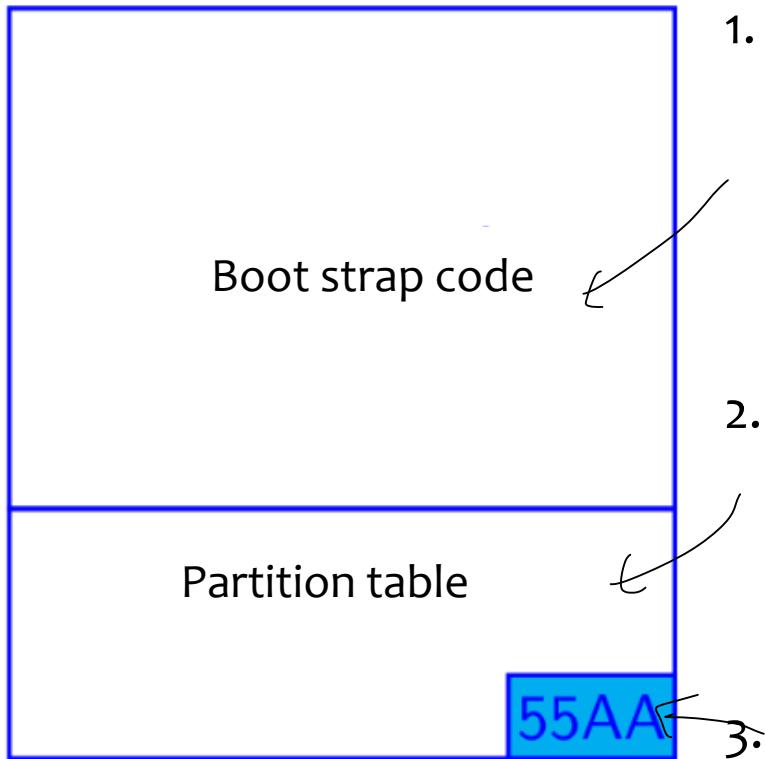
logical structure

- Floppy disk: Each OS has their own management strategy
- Hard disk (mass storage) *our focus*
 - Divided into sections (Partitions, Volumes,...)
 - Each section is a set of contiguous Cylinders
 - User can set the size (Example: use fdisk command)
 - Each partition can be managed by an independent OS
 - OS formats partition in an usable format
 - Different systems exist: FAT, NTFS, EXT3,...
 - In front of partitions are masked sector
 - Master Boot Record (MBR): Disk's first Sector
↑ Info about that disk



Master Boot Record

- Disk's most important Sector
- Disk's first sector (Number 0 or address $<0, 0, 1>$)
- Structure consist of 3 parts



1. Bootstrap code
 - Read partition table to know
 - Position of partitions
 - Active partition (contains OS)
 - Read and execute first sector of active partition
2. Partition table (64bytes)
 - Consist of 4 elements, each element 16 bytes
 - Element contains information of one partition: Position, size, owned system
3. OS's signature (always 55AA)

≡ sys signature

Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

divided into >4 partitions? → extended partition

Structure of one partition table's element

| | Stt | Ofs | Size | Meaning |
|---------------|-----|-----|------|--|
| | 1 | 0 | 1B | Active partition? 80h if yes; 0: Data |
| | 2 | 1 | 1B | Partition's first header number |
| | 2 | 2 | 1W | Partition's first sector and cylinder's number |
| Start address | 3 | | | |
| | | | | Recognition code. 05/0F: extended Partition 06:Big Dos; 07:NTFS; 0B: FAT32,.. |
| Last address | 4 | 4 | 1B | Last header's number |
| | 5 | 5 | 1B | Last sector and cylinder 's number (sector's number only use 6 lower bits) |
| | 6 | 6 | 1DW | Start address, (in sector's numbering) |
| | 7 | 8 | 1DW | Number of sectors in partition |

Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

(Little Endian format)

Example 1

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 01 | 01 | 00 | 07 | FE | 3F | F8 | 3F | 00 | 00 | 00 | 7A | 09 | 3D | 00 |
| 80 | 00 | 01 | F9 | 0B | FE | BF | 30 | B9 | 09 | 3D | 00 | 38 | 7B | 4C | 00 |
| 00 | 00 | 81 | EB | 0F | FE | FF | FF | 2B | 1D | B7 | 00 | 72 | 13 | 7A | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 55 | AA | | | | | | | | | | | | | | |

reorder

$F83F = \underbrace{00}_{2^4} \underbrace{1111}_{1000} \underbrace{1000}_{63} \underbrace{1111}_{2^{48}}$

(when decode
remember to
reverse the
order)

Decode

Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

Example 2

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 80 | 01 | 01 | 00 | 07 | FE | FF | FF | 3F | 00 | 00 | 00 | 2C | 92 | 00 | 02 |
| 00 | 00 | C1 | FF | 0F | FE | FF | FF | 31 | 41 | 8A | 03 | 0E | D3 | 1D | 01 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 55 | AA | | | | | | | | | | | | | | |

| Active | Begin | | | End | | | Relative | | Number | | |
|--------|-------|------------|-----|-----|-----|------------|----------|----------|----------|--------|--------|
| | Hdr | Cyl | Sct | Sys | Hdr | Cyl | Sct | Sector | | Of | |
| | | | | | | | | | | Sector | Sector |
| YES | 1 | 0(0) | 1 | 07 | 254 | 1023(2090) | 63 | 63 | 33591852 | | |
| NO | 0 | 1023(3697) | 1 | 0F | 254 | 1023(4862) | 63 | 59392305 | 18731790 | | |
| NO | 0 | 0(0) | 0 | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NO | 0 | 0(0) | 0 | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

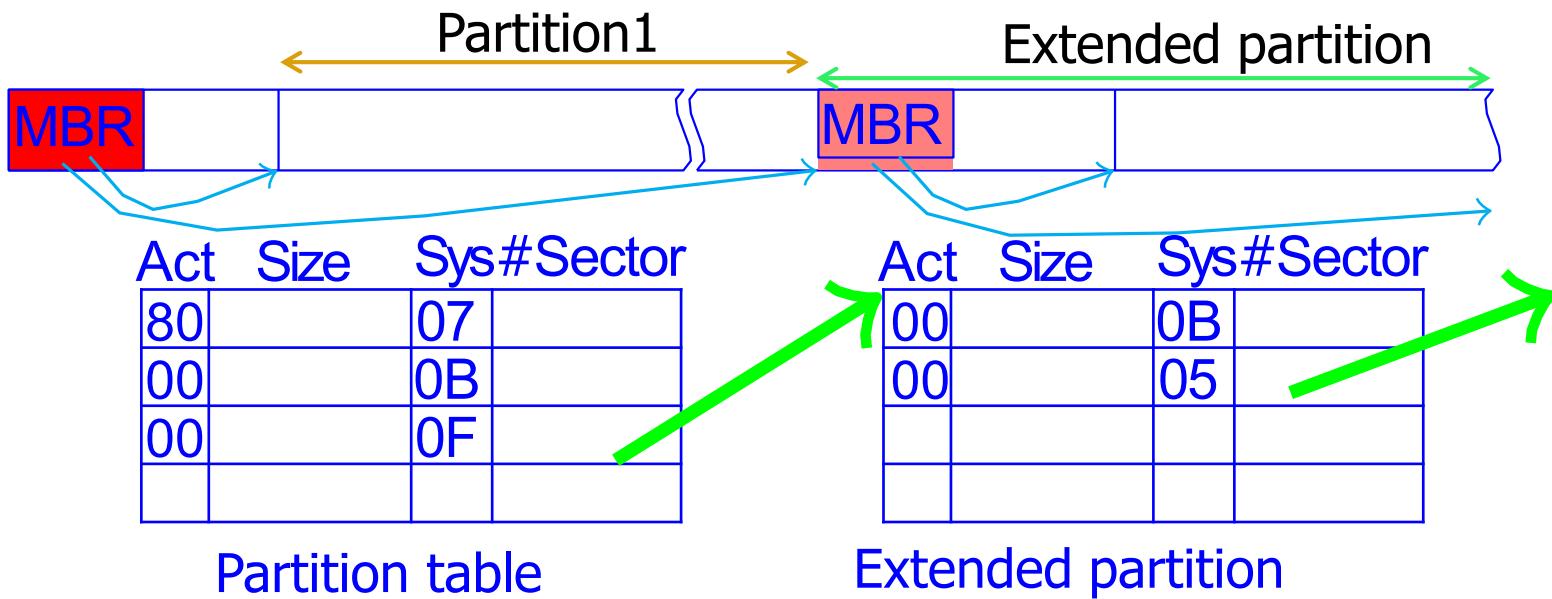
Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

Extended partition

- When recognition code is 05 or 0F -> extended partition
- Organized as a physical hard disk
 - First Sector is MBR, contain information of partitions inside this extended partition
 - Element in extended partition can be an extended partition
 - Allow to create more than 4 logic drive



Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

Example of extended partition 1

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 80 | 01 | 01 | 00 | 07 | EF | FF | FF | 3F | 00 | 00 | 00 | 11 | 2F | F7 | 01 |
| 00 | 00 | C1 | FF | 0F | EF | FF | FF | 50 | 2F | F7 | 01 | B0 | 23 | B1 | 02 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 55 | AA | | | | | | | | | | | | | | |

| Logical Block Address (LBA) Conversion | | | | | | | | | | | |
|--|-------|------|------------------|----|-----|--------------|-----|----------|-----------------------------|--------|--------|
| Sector Address | | | Cylinder Address | | | Head Address | | | Logical Block Address (LBA) | | |
| Active | Begin | End | Sys | | | | | | Relative | Number | |
| | Hdr | Cyl | Sct | | Hdr | Cyl | Sct | Sector | | | Sector |
| YES | 1 | 0 | 1 | 07 | 239 | 1023 | 63 | 63 | 32976657 | | |
| NO | 0 | 1023 | 1 | 0F | 239 | 1023 | 63 | 32976720 | 45163440 | | |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

Example of extended partition 2

Extended Partition (Sector number 32976720)...

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 01 | C1 | FF | 06 | EF | FF | FF | 3F | 00 | 00 | 00 | 51 | E8 | 76 | 01 |
| 00 | 00 | C1 | FF | 05 | EF | FF | FF | 90 | E8 | 76 | 01 | 20 | 3B | 3A | 01 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 55 | AA | | | | | | | | | | | | | | |

| Active | Begin | | | End | | | Relative | | Number | | | |
|--------|-------|------|-----|-----|-----|------|----------|----------|----------|-----|----------|--------|
| | | | | Sys | Hdr | Cyl | Sct | Hdr | Cyl | Sct | Sector | Of |
| | Hdr | Cyl | Sct | | | | | | | | | Sector |
| NO | 1 | 1023 | 1 | 06 | 239 | 1023 | 63 | | 63 | | 24569937 | |
| NO | 0 | 1023 | 1 | 05 | 239 | 1023 | 63 | 24570000 | 20593440 | | | |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |



Chapter 4: File system management

3. Information organization on disk

3.2 Disk's logical structure

Example of extended partition 3

Extended Partition (Sector number 57546720)...

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 01 | C1 | FF | 0B | EF | FF | FF | 3F | 00 | 00 | 00 | E1 | 3A | 3A | 01 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 55 | AA | | | | | | | | | | | | | | |

| Active | Begin | | | End | | | Relative | | | Number | | |
|--------|-------|------|-----|-----|-----|------|----------|--------|--------|----------|--|--|
| | Hdr | Cyl | Sct | Sys | Hdr | Cyl | Sct | Sector | Sector | Of | | |
| | | | | | | | | | | | | |
| NO | 1 | 1023 | 1 | 0B | 239 | 1023 | 63 | | 63 | 20593377 | | |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | | 0 | 0 | | |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | | 0 | 0 | | |
| NO | 0 | 0 | 0 | 00 | 0 | 0 | 0 | | 0 | 0 | | |

Chapter 4 File system management

- ① File system
- ② File system's implementation
- ③ Information organization on disk
- ④ **FAT system**

File systems

Many file systems existed

- FAT system
 - FAT 12/ FAT16 for MSDOS
 - FAT32 from WIN98
 - 12/16/32: Number of bit to identify a cluster
- NTFS
 - Used in WINNT, WIN2000
 - Utilize 64 bits to identify a cluster
 - Better than FAT in security, encoding, compress data,...
- EXT3
 - Used in Linux
- CDFS
 - File system used for CDROM
 - Limited depth in directory tree and size of names
- UDF
 - Developed from CDFS for DVD-ROM, support long file name

- Boot sector
- FAT (File Allocation Table)
- Root directory

Partitioning structure for FAT

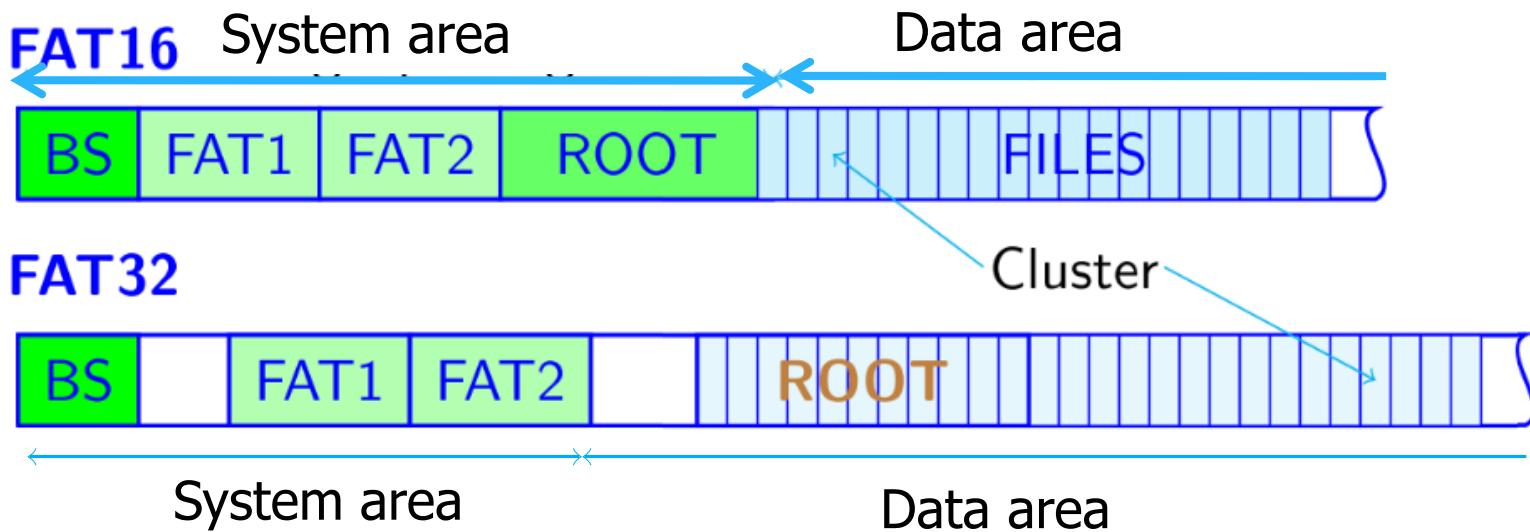
FAT12/16

- Maximum number of cluster FAT12: $2^{12} - 18$; FAT16 : $2^{16} - 18$
- Max size: FAT12: 32MB; FAT16: 2GB/4GB (32K/64K Cluster)

FAT32

- Only use 28 bits ⇒ Maximum Number of cluster: $2^{28} - 18$
- Max size: 2TB/8GB/16TB (8KB/32KB/64KB Cluster)

FAT's logical structure

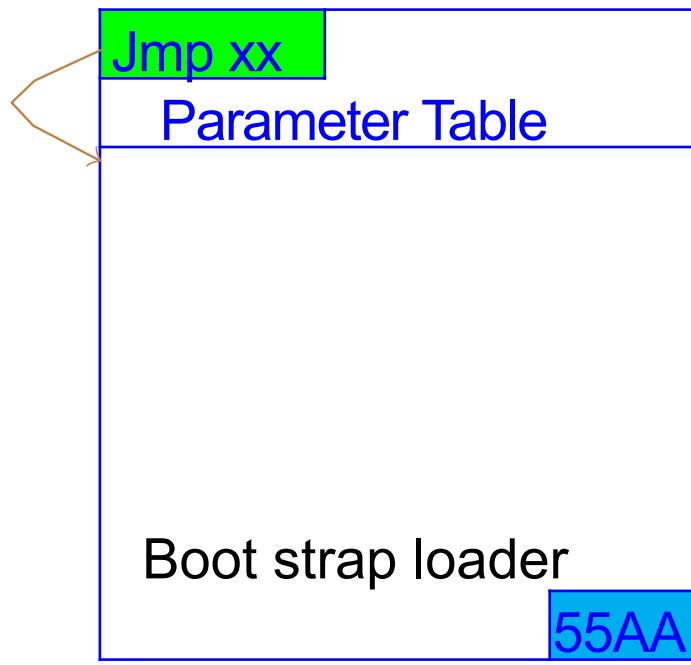


Chapter 4: File system management

4. FAT system

4.1 Boot sector

Structure



| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 58 | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 10 | 24 | 00 |
| 02 | 00 | 00 | 00 | 00 | F8 | 00 | 00 | 3F | 00 | F0 | 00 | 3F | 00 | 00 | 00 |
| E1 | 3A | 3A | 01 | 3E | 27 | 00 | 00 | 00 | 00 | 00 | 00 | 02 | 00 | 00 | 00 |
| 01 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 80 | 00 | 29 | D9 | DF | 92 | BC | 4E | 4F | 20 | 4E | 41 | 4D | 45 | 20 | 20 |
| 20 | 20 | 46 | 41 | 54 | 33 | 32 | 20 | 20 | 20 | 33 | C9 | 8E | D1 | BC | F4 |
| 7B | 8E | C1 | 8E | D9 | BD | 00 | 7C | 88 | 4E | 02 | 8A | 56 | 40 | B4 | 08 |
| CD | 13 | 73 | 05 | B9 | FF | FF | 8A | F1 | 66 | 0F | B6 | C6 | 40 | 66 | 0F |
| B6 | D1 | 80 | E2 | 3F | F7 | E2 | 86 | CD | C0 | ED | 06 | 41 | 66 | 0F | B7 |
| C9 | 66 | F7 | E1 | 66 | 89 | 46 | F8 | 83 | 7E | 16 | 00 | 75 | 38 | 83 | 7E |
| 2A | 00 | 77 | 32 | 66 | 8B | 46 | 1C | 66 | 83 | C0 | 0C | BB | 00 | 80 | B9 |
| 01 | 00 | E8 | 2B | 00 | E9 | 48 | 03 | A0 | FA | 7D | B4 | 7D | 8B | F0 | AC |
| 84 | C0 | 74 | 17 | 3C | FF | 74 | 09 | B4 | 0E | BB | 07 | 00 | CD | 10 | EB |
| EE | A0 | FB | 7D | EB | E5 | A0 | F9 | 7D | EB | E0 | 98 | CD | 16 | CD | 19 |
| 66 | 60 | 66 | 3B | 46 | F8 | 0F | 82 | 4A | 00 | 66 | 6A | 00 | 66 | 50 | 06 |
| 53 | 66 | 68 | 10 | 00 | 01 | 00 | 80 | 7E | 02 | 00 | 0F | 85 | 20 | 00 | B4 |
| 41 | BB | AA | 55 | 8A | 56 | 40 | CD | 13 | 0F | 82 | 1C | 00 | 81 | FB | 55 |
| AA | 0F | 85 | 14 | 00 | F6 | C1 | 01 | 0F | 84 | 0D | 00 | FE | 46 | 02 | B4 |
| 42 | 8A | 56 | 40 | 8B | F4 | CD | 13 | B0 | F9 | 66 | 58 | 66 | 58 | 66 | 58 |
| 66 | 58 | EB | 2A | 66 | 33 | D2 | 66 | 0F | B7 | 4E | 18 | 66 | F7 | F1 | FE |
| C2 | 8A | CA | 66 | 8B | D0 | 66 | C1 | EA | 10 | F7 | 76 | 1A | 86 | D6 | 8A |
| 56 | 40 | 8A | E8 | C0 | E4 | 06 | 0A | CC | B8 | 01 | 02 | CD | 13 | 66 | 61 |
| 0F | 82 | 54 | FF | 81 | C3 | 00 | 02 | 66 | 40 | 49 | 0F | 85 | 71 | FF | C3 |
| 4E | 54 | 4C | 44 | 52 | 20 | 20 | 20 | 20 | 20 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 6D | 6F | 76 | 65 | 20 | 64 | 69 | 73 | 6B | 73 | 20 | 6F | 72 | 20 | 6F | 74 |
| 68 | 65 | 72 | 20 | 6D | 65 | 64 | 69 | 61 | 2E | FF | 0D | 0A | 44 | 69 | 73 |
| 6B | 20 | 65 | 72 | 72 | 6F | 72 | FF | 0D | 0A | 50 | 72 | 65 | 73 | 73 | 20 |
| 61 | 6E | 79 | 20 | 6B | 65 | 79 | 20 | 74 | 6F | 20 | 72 | 65 | 73 | 74 | 61 |
| 72 | 74 | 0D | 0A | 00 | 00 | 00 | 00 | 00 | AC | CB | D8 | 00 | 00 | 55 | AA |

- Partition's First Sector
- Structure consist of 3 parts
 - BPB: Bios Parameter Block
 - Boot strap loader
 - System's signature (always 55AA)

Chapter 4: File system management

4. FAT system

4.1 Boot sector

Bios Parameter Block's structure – Common part

| Stt | Ofs | Size | Sample value | Meaning |
|-----|-----|------|--------------|---|
| 1 | 0 | 3B | EB 3C 90 | Jump to begin position of boot strap loader |
| 2 | 3 | 8B | MSDOS5.0 | Name of system used formatted the disk |
| 3 | 11 | 1W | 00 02 | Size of 1 sector, normally 512 |
| 4 | 13 | 1B | 40 | Number of sectors for 1 cluster (32K-Cluster) |
| 5 | 14 | 1W | 01 00 | Num of scts before FAT/Num of reserved scts |
| 6 | 16 | 1B | 02 | Num of FAT |
| 7 | 17 | 1W | 00 02 | Num of ROOT's element. FAT32: 00 00 |
| 8 | 19 | 1W | 00 00 | Total sector on disk (< 32M) or 0000 |
| 9 | 21 | 1B | F8 | Disk format (F8:HD, F0: Disk 1.44M) |
| 10 | 22 | 1W | D1 00 | Num of sector for one FAT(209) |
| 11 | 24 | 1W | 3F 00 | Num of sector for one track (63) |
| 12 | 26 | 1W | 40 00 | Num of headers (64) |
| 13 | 28 | 1DW | 3F 00 00 00 | Num of hidden sector – Sectors before volume (63) |
| 14 | 32 | 1DW | 41 0C 34 00 | Total sector on disk (3411009) |

Chapter 4: File system management

4. FAT system

4.1 Boot sector

Bios Parameter Block's structure – Part for FAT12/FAT16 only

| Stt | Ofs | Kt | Sample value | Meaning |
|-----|-----|-----|--------------|---|
| 15 | 36 | 1B | 80h | Physical disk num 0: A drive; 80h: C drive |
| 16 | 37 | 1B | 00 | Reserved, high byte is for disk number |
| 17 | 38 | 1B | 29h | Boot sector extension 29h |
| 18 | 39 | 1DW | D513 5B24 | Volume Serial number(245B-13D5) |
| 19 | 43 | 11B | NO NAME | Volume's Label |
| 20 | 54 | 8B | FAT16 | Reserved, normally for FAT description's text |
| 21 | 62 | - | | Bootstrap loader |

Example:

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

Chapter 4: File system management

4. FAT system

4.1 Boot sector

FAT16's disk parameter decoding example

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

First 3 Bytes đầu: Jump to the start of boot strap loader

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Jmp+3C

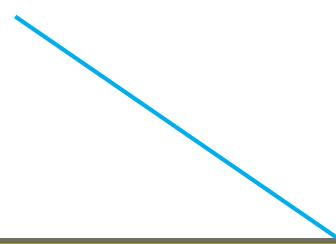
| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

8 Bytes

Name of the system utilized to format disk

OENName: MSDOS5.0



| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

1 Word

Size of 1 sector

Sector's size: 512

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of sector for 1 cluster

2 sector for 1 cluster

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of sector before FAT

6 sectors before the first FAT

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

There are 2 FAT

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of ROOT's elements

There are maximum 512 elements in Root's directory

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Total sector on disk

Disk larger than 32MB

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Disk format code: F8

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of sector for 1 FAT:245

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of sector for 1 track: 63

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of headers: 255

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Number of hidden sectors: 63

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Total sector of Volume: 125889(\approx 64MB)

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Physical disk number: 00 00

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Bootsector extension: 29h

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Volume serial number: 70D4-EAA6

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Disk label: NO NAME

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

FAT Type: FAT16

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

FAT16's disk parameter decoding example

Start of boot strap loader

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 3C | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 02 | 06 | 00 |
| 02 | 00 | 02 | 00 | 00 | F8 | F5 | 00 | 3F | 00 | FF | 00 | 3F | 00 | 00 | 00 |
| C1 | EB | 01 | 00 | 00 | 00 | 29 | A6 | EA | D4 | 70 | 4E | 4F | 20 | 4E | 41 |
| 4D | 45 | 20 | 20 | 20 | 20 | 46 | 41 | 54 | 31 | 36 | 20 | 20 | 20 | 33 | C9 |

Bios Parameter Block's structure – Part for FAT32

| Stt | Ofs | Size | Sample value | Meaning |
|-----|-----|------|--------------|---|
| 15 | 36 | 1DW | C9 03 00 00 | Total sector for FAT |
| 16 | 40 | 1W | 00 00 | Flags: main #FAT (not used) |
| 17 | 42 | 1W | 00 00 | Version: FAT32 (<i>not used</i>) |
| 18 | 44 | 1DW | 02 00 00 00 | cluster starting number of ROOT |
| 19 | 48 | 1W | 01 00 | #sector contain File System information |
| 20 | 50 | 1W | 06 00 | Number of sector to backup Bootsector |
| 21 | 52 | 12B | 00 . . . 00 | Reserved |
| 22 | 64 | 1B | 00 | Physical disk number 0: drive A; 80h: drive C |
| 23 | 65 | 1B | 00 | Reserved/High Byte for #Driver |
| 24 | 66 | 1B | 29 | Boot sector extension. Always 29h |
| 25 | 67 | 1DW | 62 0E 18 66 | Volume Serial number |
| 26 | 71 | 11B | NO NAME | Volume Label: (<i>not used</i>) |
| 27 | 82 | 8B | FAT32 | Reserved, for FAT's description text |

Chapter 4: File system management

4. FAT system

4.1 Boot sector

FAT32 Boot sector example

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| EB | 58 | 90 | 4D | 53 | 44 | 4F | 53 | 35 | 2E | 30 | 00 | 02 | 10 | 24 | 00 |
| 02 | 00 | 00 | 00 | 00 | F8 | 00 | 00 | 3F | 00 | F0 | 00 | 3F | 00 | 00 | 00 |
| E1 | 3A | 3A | 01 | 3E | 27 | 00 | 00 | 00 | 00 | 00 | 00 | 02 | 00 | 00 | 00 |
| 01 | 00 | 06 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 80 | 00 | 29 | D9 | DF | 92 | BC | 4E | 4F | 20 | 4E | 41 | 4D | 45 | 20 | 20 |
| 20 | 20 | 46 | 41 | 54 | 33 | 32 | 20 | 20 | 20 | 33 | C9 | 8E | D1 | BC | F4 |
| 7B | 8E | C1 | 8E | D9 | BD | 00 | 7C | 88 | 4E | 02 | 8A | 56 | 40 | B4 | 08 |
| CD | 13 | 73 | 05 | B9 | FF | FF | 8A | F1 | 66 | 0F | B6 | C6 | 40 | 66 | 0F |
| B6 | D1 | 80 | E2 | 3F | F7 | E2 | 86 | CD | C0 | ED | 06 | 41 | 66 | 0F | B7 |
| C9 | 66 | F7 | E1 | 66 | 89 | 46 | F8 | 83 | 7E | 16 | 00 | 75 | 38 | 83 | 7E |
| 2A | 00 | 77 | 32 | 66 | 8B | 46 | 1C | 66 | 83 | C0 | 0C | BB | 00 | 80 | B9 |
| 01 | 00 | E8 | 2B | 00 | E9 | 48 | 03 | A0 | FA | 7D | B4 | 7D | 8B | F0 | AC |
| 84 | C0 | 74 | 17 | 3C | FF | 74 | 09 | B4 | 0E | BB | 07 | 00 | CD | 10 | EB |
| EE | A0 | FB | 7D | EB | E5 | A0 | F9 | 7D | EB | E0 | 98 | CD | 16 | CD | 19 |
| 66 | 60 | 66 | 3B | 46 | F8 | 0F | 82 | 4A | 00 | 66 | 6A | 00 | 66 | 50 | 06 |
| 53 | 66 | 68 | 10 | 00 | 01 | 00 | 80 | 7E | 02 | 00 | 0F | 85 | 20 | 00 | B4 |
| 41 | BB | AA | 55 | 8A | 56 | 40 | CD | 13 | 0F | 82 | 1C | 00 | 81 | FB | 55 |
| AA | 0F | 85 | 14 | 00 | F6 | C1 | 01 | 0F | 84 | 0D | 00 | FE | 46 | 02 | B4 |
| 42 | 8A | 56 | 40 | 8B | F4 | CD | 13 | B0 | F9 | 66 | 58 | 66 | 58 | 66 | 58 |
| 66 | 58 | EB | 2A | 66 | 33 | D2 | 66 | 0F | B7 | 4E | 18 | 66 | F7 | F1 | FE |
| C2 | 8A | CA | 66 | 8B | D0 | 66 | C1 | EA | 10 | F7 | 76 | 1A | 86 | D6 | 8A |
| 56 | 40 | 8A | E8 | C0 | E4 | 06 | 0A | CC | B8 | 01 | 02 | CD | 13 | 66 | 61 |
| 0F | 82 | 54 | FF | 81 | C3 | 00 | 02 | 66 | 40 | 49 | 0F | 85 | 71 | FF | C3 |
| 4E | 54 | 4C | 44 | 52 | 20 | 20 | 20 | 20 | 20 | 20 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 0D | 0A | 52 | 65 |
| 6D | 6F | 76 | 65 | 20 | 64 | 69 | 73 | 6B | 73 | 20 | 6F | 72 | 20 | 6F | 74 |
| 68 | 65 | 72 | 20 | 6D | 65 | 64 | 69 | 61 | 2E | FF | 0D | 0A | 44 | 69 | 73 |
| 6B | 20 | 65 | 72 | 72 | 6F | 72 | FF | 0D | 0A | 50 | 72 | 65 | 73 | 73 | 20 |
| 61 | 6E | 79 | 20 | 6B | 65 | 79 | 20 | 74 | 6F | 20 | 72 | 65 | 73 | 74 | 61 |
| 72 | 74 | 0D | 0A | 00 | 00 | 00 | 00 | 00 | AC | CB | D8 | 00 | 00 | 55 | AA |



FAT32 boot sector decoding result

| BIOS PARAMETER BLOCK (BPB)... | |
|---------------------------------------|-------------------|
| OEM Name | : MSDOS5.0 |
| Bytes per sector | : 512 |
| Sectors per cluster | : 16 |
| Sectors before the first FAT | : 36 |
| Number of copies of FAT | : 2 |
| Media Desctiptor | : F8h |
| Sectors per Tracks | : 63 |
| Number of Header | : 240 |
| Number of Hiden Scts in Volume | : 63 |
| Number of Sectors in Volume | : 20593377 |
| Number of Sectors per FAT | : 10046 |
| Cluster num. of start of ROOT | : 2 |
| Sct number of FileSystem Info | : 1 |
| Sct number of Boot backup sct | : 6 |
| Logical drive number of Volume | : 80h |
| Extend BPB Signature | : 29h |
| Serial Number of Volume | : BC92-DFD9 |
| Volume label | : NO NAME |
| FAT Type | : FAT32 |
| Boot signature | : 55 AA |

File System Information Sector

- Usually Sector # 2 of Volume
 - Right after Boot sector (Sector # 1)
- Structure

| Stt | Ofs | Size | Meaning |
|-----|-----|------|---|
| 1 | 0 | 1DW | First signature of FSInfo sector. Bytes' values in order: 52h 52h 61h 41h |
| 2 | 4 | 480B | Not used, usually contain value 00 |
| 3 | 484 | 1DW | Signature of File System Information Sector. Bytes values in order: 72h 72h 41h 61h |
| 4 | 488 | 1DW | Number of free cluster. -1 if not defined |
| 5 | 492 | 1DW | Number of cluster just provided |
| 6 | 496 | 12B | Reserved |
| 7 | 508 | 2B | Not defined, usually 0 |
| 8 | 510 | 2B | Bootsector's signature. Contain value 55 AA |

Chapter 4: File system management

4. FAT system

4.1 Boot sector

File system information sector of 1volume use FAT32

File system information sector of 1 volume use FAT32

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 52 | 52 | 61 | 41 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |

FILE SYSTEM INFO:...

| | | |
|-------------------------------------|---|-----------|
| First signature | : | 41615252h |
| File System Info Signature | : | 61417272h |
| Number of Free Clusters | : | 786100 |
| #Cluster recently Allocation | : | 2469 |
| Boot signature | : | 55 AA |

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 72 | 72 | 41 | 61 | B4 | FE | 0B | 00 | A5 | 09 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 55 | AA | | | | | | | | | | | |

- Boot sector
- **FAT (File Allocation Table)**
- Root directory

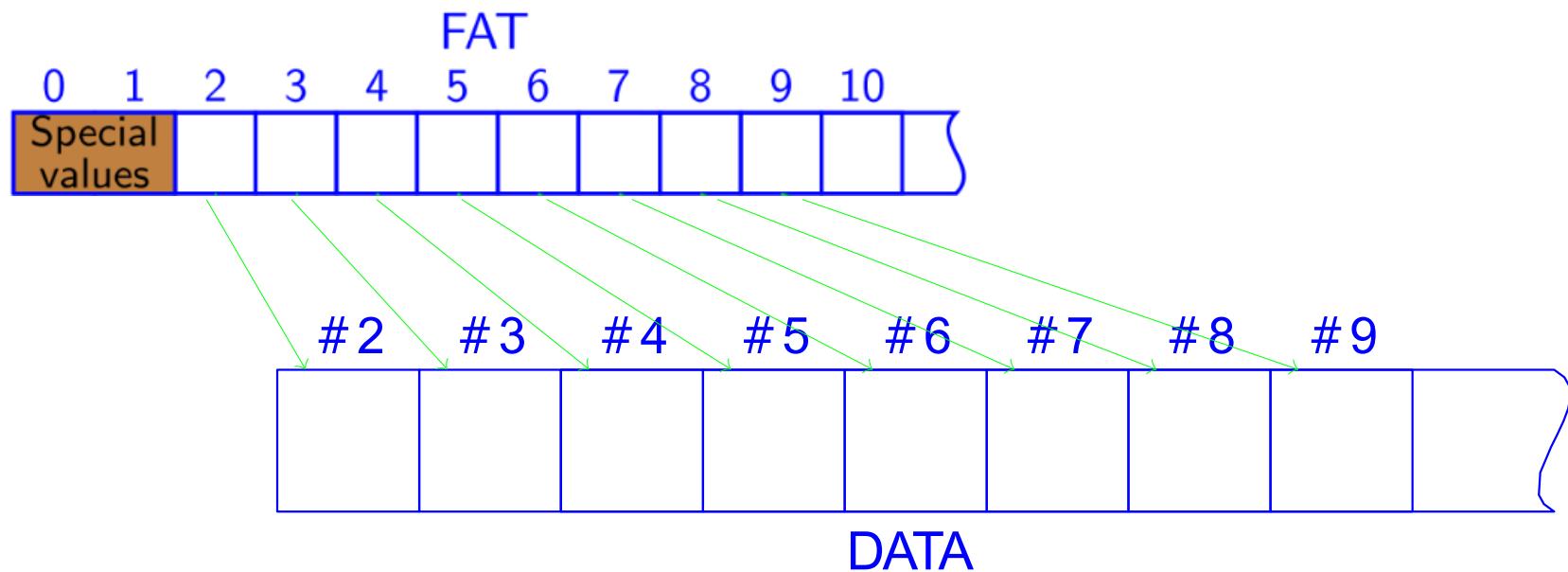
FAT utilized to manage memory blocks/clusters in the data area of storage memory

- Using block
 - Allocated to each file/directory
- Free block
- Error block

Method

FAT include many elements

- Each element can be 12bit, 16bit, 32bit
 - Each element corresponding to 1 block (cluster) on the data area
 - First 2 elements (0,1) has special meaning
 - Disk format, Bit shutdown, Bit diskerror
 - Element # 2 corresponding to first cluster of the Data



Implementation

Each element of FAT contain a value represents the property of corresponding cluster

| FAT[(32)16]12 | Meaning |
|--|--|
| [(0000)0]000h | Cluster is free |
| [(0000)0]001h | Unused value |
| [(0000)0]002h →[(0FFF)F]FEFh | Cluster being used. Value is a pointer, point to next cluster of file |
| [(0FFF)F]FF0h →[(0FFF)F]FF6h | Reserved value, unused |
| [(0FFF)F]FF7h | Mark corresponding cluster is broken |
| [(0FFF)F]FF8h→ →[(0FFF)F]FFFh | Cluster is being used and it's the last cluster of file (<i>EOC:End Of Cluster chain</i>). Usually value: [(0FFF)F]FFFh |

Chapter 4: File system management

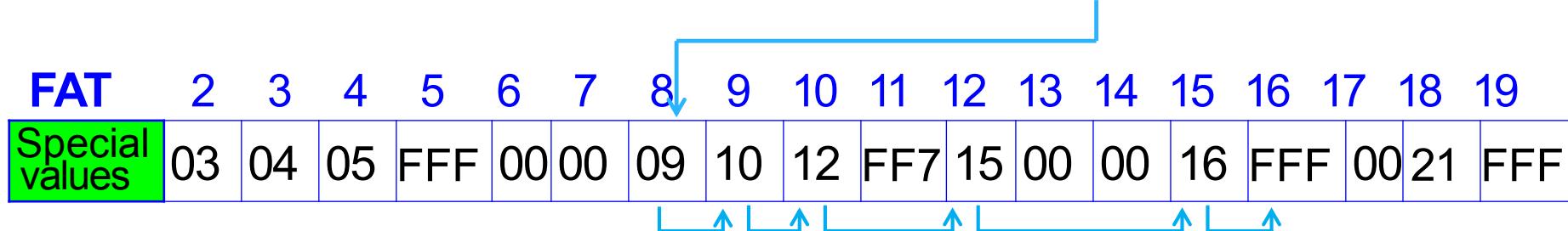
4. FAT system

4.2 FAT

Cluster linking

Root entry

| | | | | | | | |
|-----|-----|---|--|------|------|-----|------|
| ABC | TXT | A | | Time | Date | 008 | Size |
|-----|-----|---|--|------|------|-----|------|



| | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|

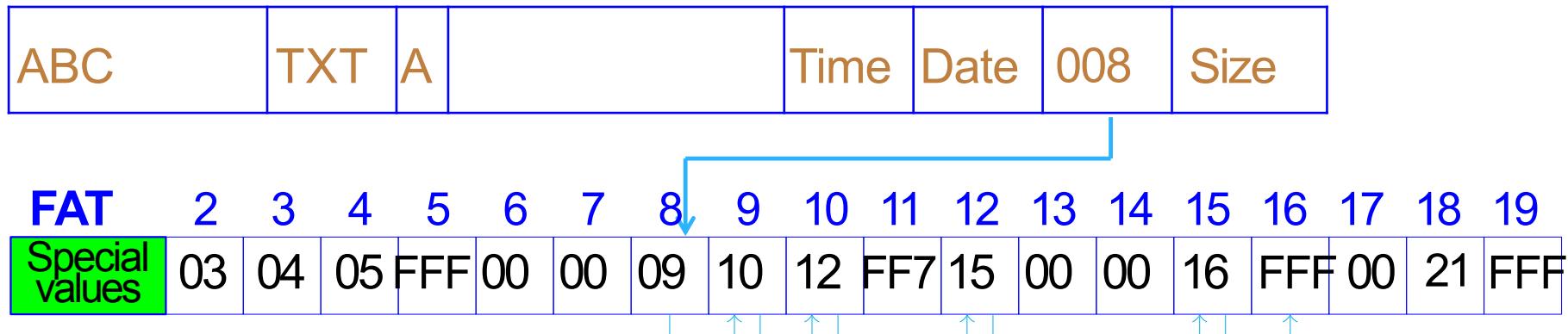
#2 #3 #4 #5 #6 #7 #8 #9 #10 #11

| | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|
| | | | | | | | | | | |
|--|--|--|--|--|--|--|--|--|--|--|

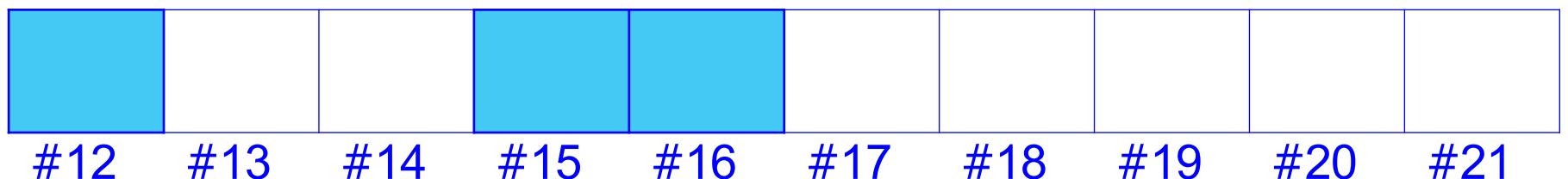
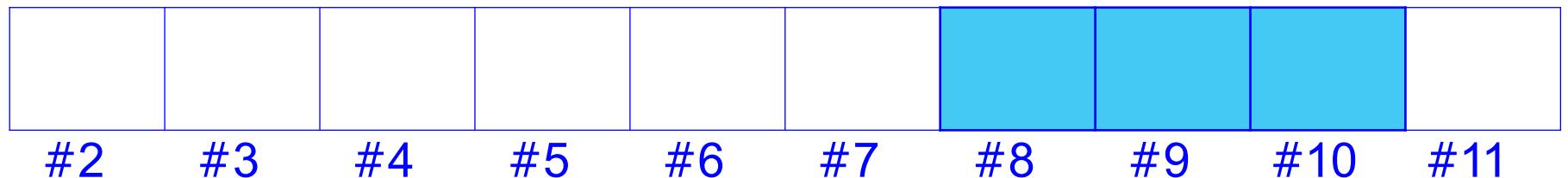
#12 #13 #14 #15 #16 #17 #18 #19 #20 #21

Cluster linking

Root entry



DATA



Chapter 4: File system management

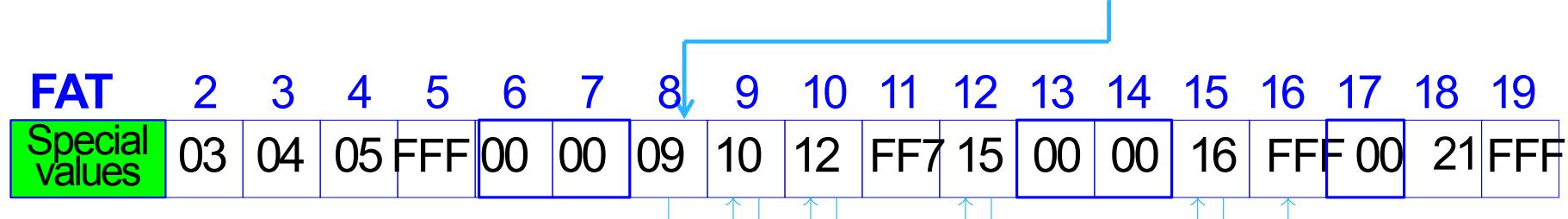
4. FAT system

4.2 FAT

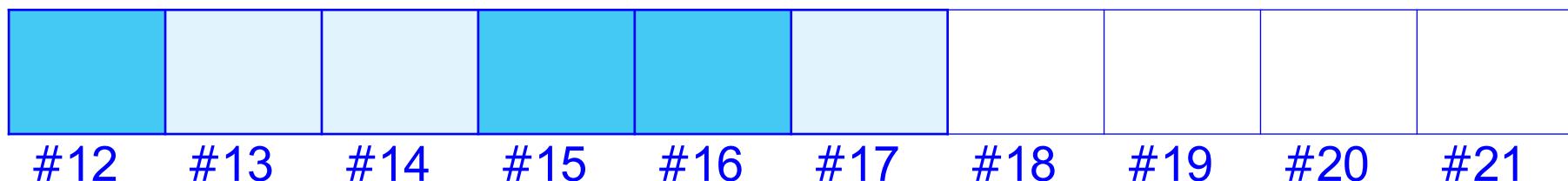
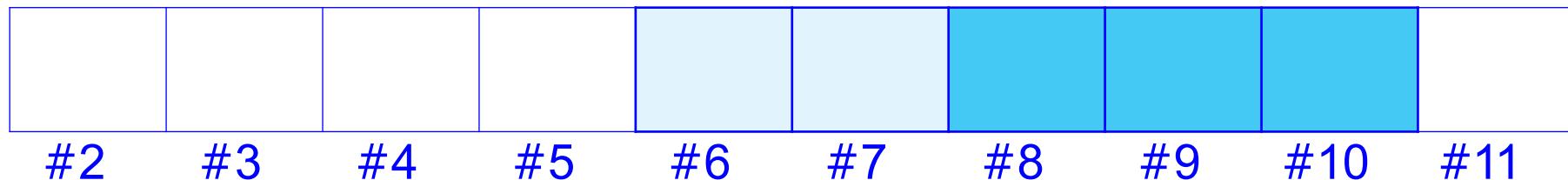
Cluster linking

Root entry

| | | | | | | | |
|-----|-----|---|--|------|------|-----|------|
| ABC | TXT | A | | Time | Date | 008 | Size |
|-----|-----|---|--|------|------|-----|------|



DATA



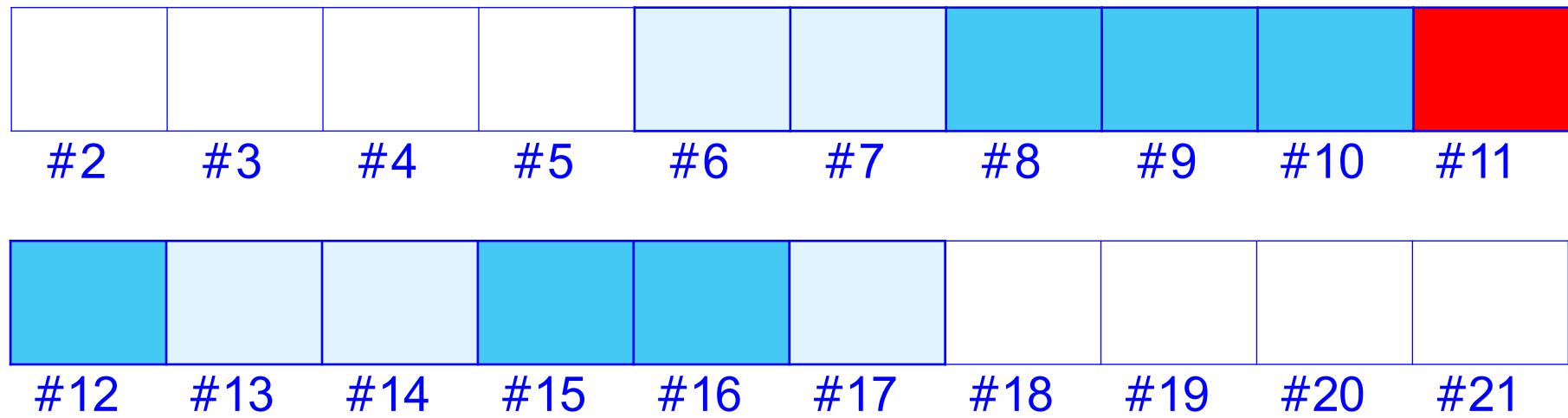
Cluster linking

Root entry

The diagram illustrates the file structure and its representation in the FAT table. The top row shows the file entries: ABC, TXT, A, (empty), Time, Date, 008, and Size. Below this, the FAT table is shown with entries from 2 to 19. The entry at index 8 contains the value FF7, which is highlighted with a red box and a blue arrow pointing from the 'Time' field in the file structure. The entry at index 12 also contains FF7 and is highlighted with a red box and a blue arrow pointing from the 'Date' field in the file structure. The entry at index 15 contains 00, which is highlighted with a green box and a blue arrow pointing from the 'Size' field in the file structure. The entry at index 16 contains FFF, which is highlighted with a green box and a blue arrow pointing from the 'Time' field in the file structure. The entry at index 17 contains 00, which is highlighted with a green box and a blue arrow pointing from the 'Date' field in the file structure. The entry at index 19 contains FFF, which is highlighted with a green box and a blue arrow pointing from the 'Size' field in the file structure. The entry at index 20 contains 21, which is highlighted with a green box and a blue arrow pointing from the 'Size' field in the file structure. The entry at index 21 contains FFF, which is highlighted with a green box and a blue arrow pointing from the 'Time' field in the file structure.

| | | | | | | | | | | | | | | | | | | |
|----------------|-----|----|----|------|------|-----|------|----|----|-----|----|----|----|----|-----|----|----|-----|
| ABC | TXT | A | | Time | Date | 008 | Size | | | | | | | | | | | |
| FAT | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
| Special values | 03 | 04 | 05 | FFF | 00 | 00 | 09 | 10 | 12 | FF7 | 15 | 00 | 00 | 16 | FFF | 00 | 21 | FFF |

DATA



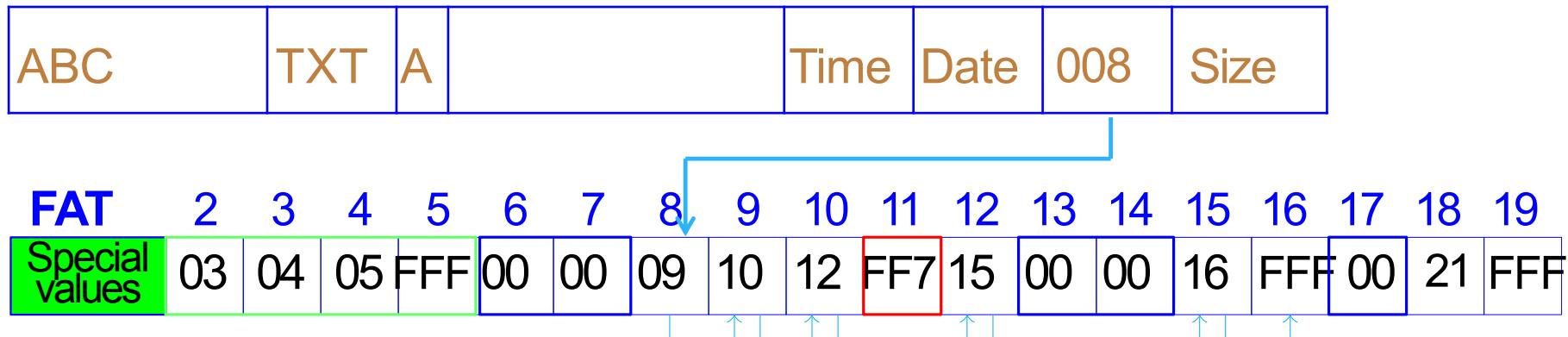
Chapter 4: File system management

4. FAT system

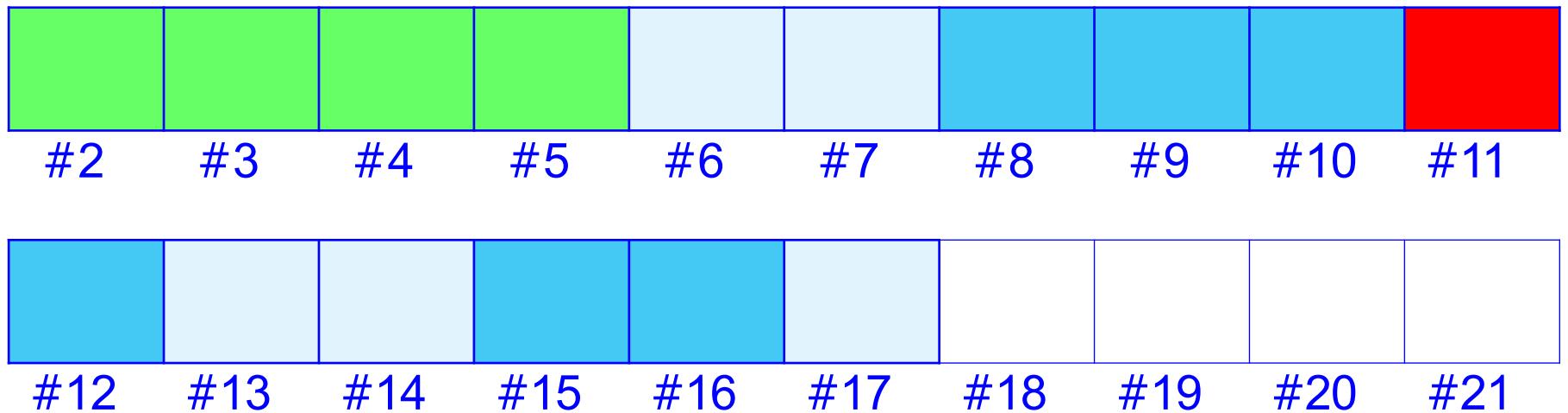
4.2 FAT

Cluster linking

Root entry



DATA



Example: Read one sector of FAT32

```
#include <windows.h>
#include <stdio.h>
int main(int argc, char *argv[]){
    HANDLE hDisk;
    BYTE Buf[512]; DWORD FAT[128];
    WORD FATAddr; DWORD byteread, i;
    hDisk = CreateFile("\\\\.\\"F:", GENERIC_READ,
                      FILE_SHARE_READ|FILE_SHARE_WRITE, NULL,
                      OPEN_EXISTING,0,NULL);
    ReadFile(hDisk,Buf,512,&byteread,NULL);
    memcpy(&FATAddr,&Buf[14],2); //Offset 14 Sector truoc FAT
    SetFilePointer(hDisk,FATAddr * 512, NULL,FILE_BEGIN);
    ReadFile(hDisk,FAT,512,&byteread,NULL);
    for(i=0;i<128;i++) printf(" %08X ",FAT[i]);
    CloseHandle(hDisk);
    return 0;
}
```

Chapter 4: File system management

4. FAT system

4.2 FAT

Example: First Sector of a FAT32

A Root entry

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 52 | 45 | 41 | 44 | 4D | 42 | 52 | 20 | 43 | 20 | 20 | 20 | 00 | 5E | B9 | 5A |
| A8 | 3E | A8 | 3E | 00 | 00 | CE | 79 | A4 | 3E | 03 | 00 | BD | 0A | 00 | 00 |

FAT

- Boot sector
- FAT (File Allocation Table)
- Root directory

Structure of root directory

- Table consists of **file record**
 - Each record size is 32 bytes
 - Contain information involve one file/directory/ volume label
- FAT12/FAT16
 - Root directory lie right behind FAT
 - Size = Number of maximum elements in root directory * 32
- FAT32
 - Location is defined based on BPB
 - Filed #18: Number of ROOT's first cluster
 - Undefined size
 - Support long file name (LFN: Long File Name)
 - One file may use more than 1 element

Structure of one element

| Stt | Ofs | Size | Meaning |
|-----|-----|------|---|
| 1 | 0 | 8B | File name |
| 2 | 8 | 3B | Extension |
| 3 | 11 | 1B | File's attribute |
| 4 | 12 | 10B | Not used in FAT12/FAT16. Used in FAT32 |
| 4.1 | 12 | 1B | Reserved |
| 4.2 | 13 | 1B | File created time, unit 10ms |
| 4.3 | 14 | 1W | File created time (<i>hour - minute - second</i>) |
| 4.4 | 16 | 1W | File created date |
| 4.5 | 18 | 1W | Last access date |
| 4.6 | 20 | 1W | Number of starting cluster of file (FAT32: <i>High part</i>) |
| 5 | 22 | 1W | Last update time |
| 6 | 24 | 1W | Last update date |
| 7 | 26 | 1W | Number of starting cluster of file (FAT32: <i>Low part</i>) |
| 8 | 28 | 1DW | Size in byte |

Structure of an element : file name

- Sequence of ASCII contain file name
- Not accept space in file name
 - Command copy, del,... do not recognize name contains space
- If smaller than 8 character, insert space character for sufficient 8 character
- First character may contain special meaning
 - 00h: First character of not used part
 - E5h (character "δ"): File corresponding to this element was deleted
 - 2Eh (character ".") : Sub directory
 - cluster number field point to itself
 - Structure of sub directory is similar to root directory: contain 32bytes elements

Structure of an element : file name

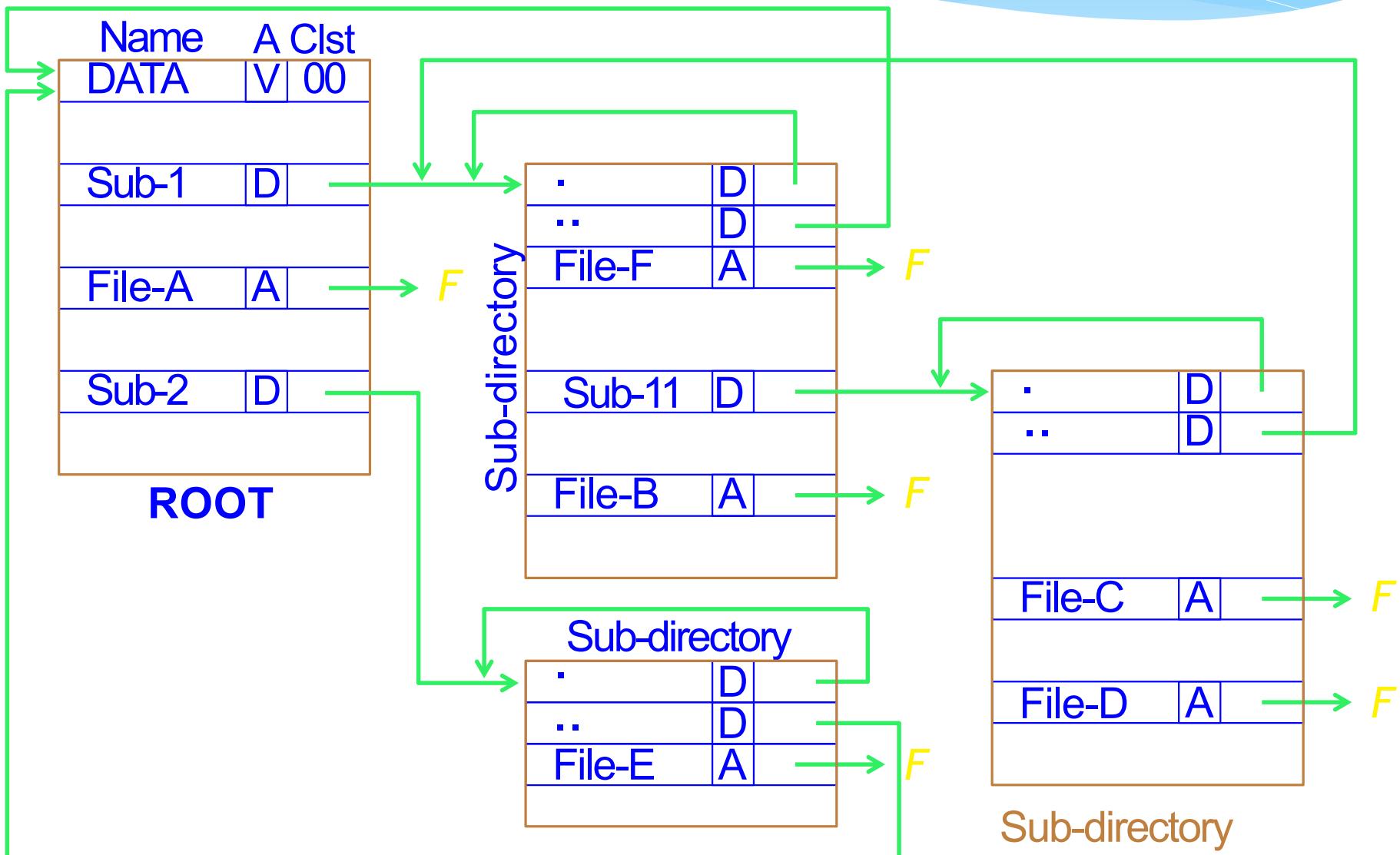
- First character may contain special meaning (**continues**)
 - 2Eh2Eh (character ".."): Parent directory of current directory
 - Cluster number field point to parent directory
 - If parent is root, #cluster start by zero (FAT12/16)
 - Sub directory lie in **Data area**, manage similar to a file
⇒ **File of file record**
 - FAT12/16: Root directory is at a defined position;
FAT32: Root directory lies in **Data area**

Chapter 4: File system management

4. FAT system

4.3 Root directory

Sub directory

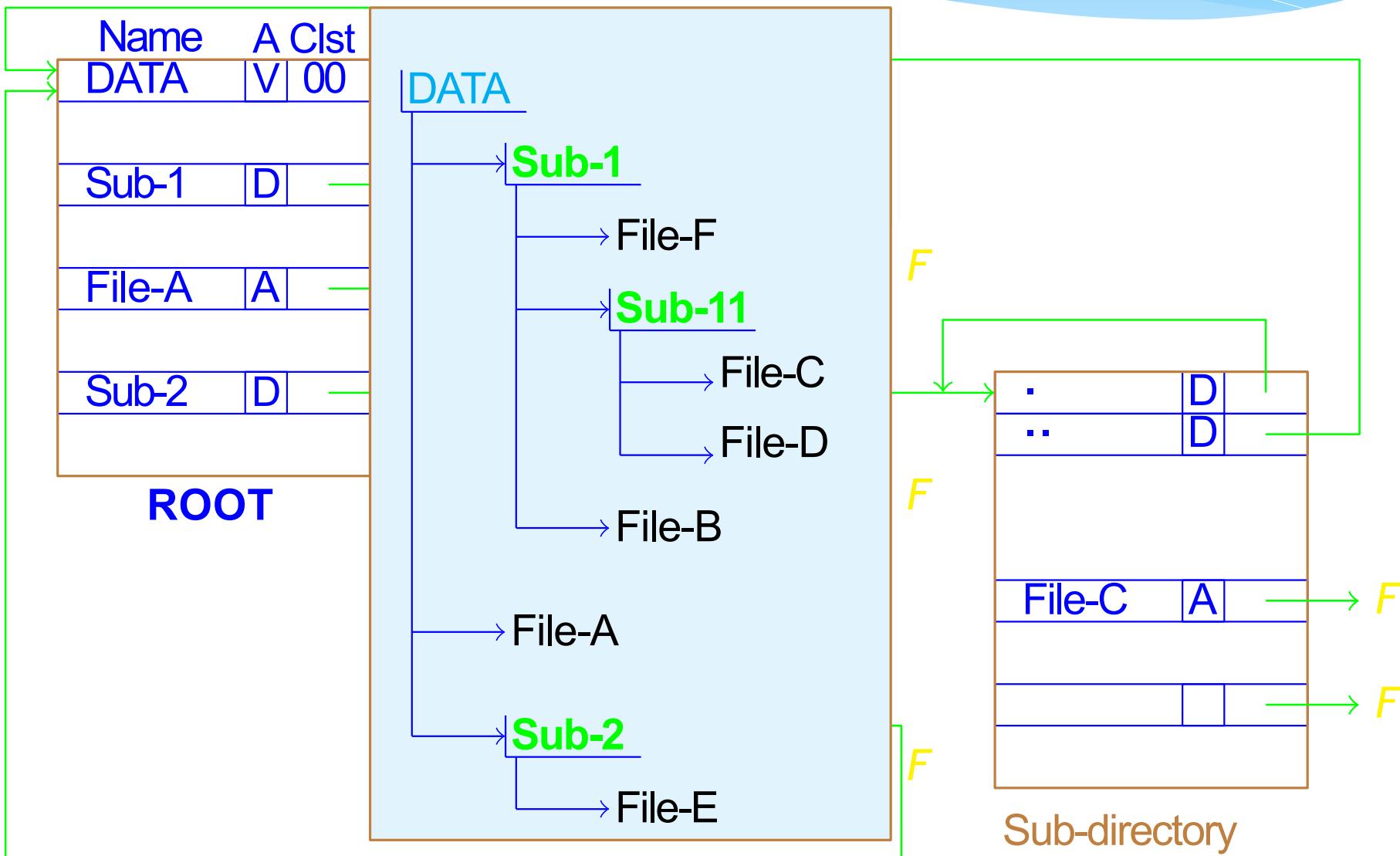


Chapter 4: File system management

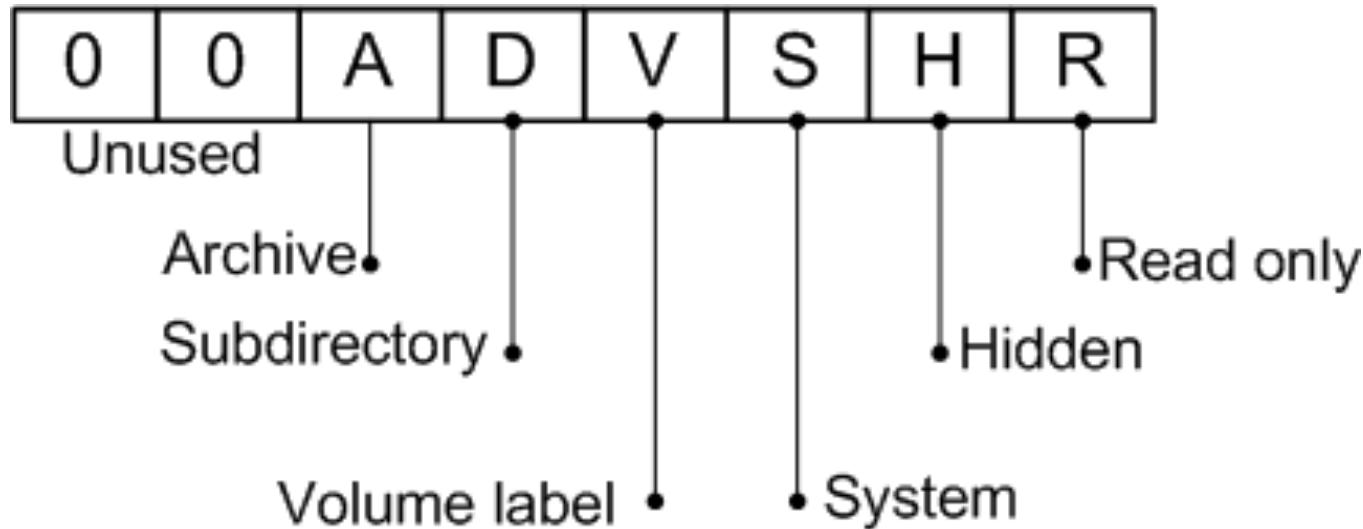
4. FAT system

4.3 Root directory

Sub directory



Structure of an element : Attribute field



Example: Byte attribute **0Fh**:

⇒ File has attribute **Volume label+System+Hidden+Read only**

Note: Attribute byte 's value **0x0F** is not used in *MS-DOS* ⇒ For marking the *Long File Name element*

Structure of an element



Example: 15 hour 34 minute 45 second

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

Value: **7C56**

Structure of an element : Date

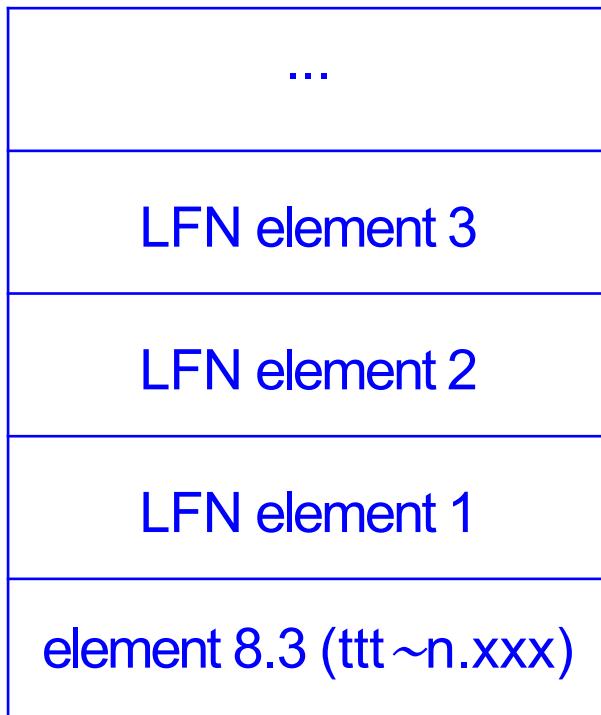


Example: Day 17 month 5 Year 2011

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |
|---|---|---|---|---|---|---|--|---|---|---|---|---|---|---|---|---|

Value : **3EB1**

Long File Name (LFN) system



| Ofs | Size | Meaning |
|-----|------|--|
| 0 | 1B | Element Order |
| 1 | 5W | First 5 unicode characters |
| 11 | 1B | Attribute. Mark <i>LFN element</i> Always value 0Fh |
| 12 | 1B | Reserved (00) |
| 13 | 1B | Checksum: Allow check if long file name is corresponding to 8.3 name? |
| 14 | 6W | unicode character 6,7,8,9,10,11 |
| 26 | 1W | Cluster number. Not used (0000) |
| 28 | 1W | unicode character 12 |
| 30 | 1W | unicode character 13 |

Long File Name (LFN) system: Ordering field

- Show the order of LFN element
 - Each LFN element contain 13 Unicode characters
- First element has value of ordering field: 1
- Last element use bit number 6 to mark
 - Only use maximum 20 elements
 - After last character is 0x00 0x00.
 - Unused character has values 0xFF 0xFF
- Bit number 7 (0x80) show corresponding element is deleted
- Example: file "This is a very long file name.docx"

| Entry | Ord | Attr | Data |
|----------|--------------|------|---------------|
| LFN 3 | 0x43 | 0x0F | ame.docx |
| LFN 2 | 0x02 | 0x0F | y long file n |
| LFN 1 | 0x01 | 0x0F | This is a ver |
| 8.3 Name | THISIS~1.DOC | | |

Chapter 4: File system management

4. FAT system

4.3 Root directory

Example: A sector of ROOT

Example: Content of ROOT

```
F:\>DIR
```

Volume in drive F is DATA

Volume Serial Number is DC27-F353

Directory of F:\

| | | | |
|------------|----------|-----------|----------------------------------|
| 05/05/2011 | 06:36 AM | <DIR> | Exemples |
| 04/26/2011 | 11:35 AM | | 465 ReadBiosSector.c |
| 05/04/2011 | 03:14 PM | | 2,749 READMBR.C |
| 05/05/2011 | 06:52 PM | <DIR> | Temps |
| 05/05/2011 | 06:35 AM | | 2,696,504 Bài gi?ng chuong 4.pdf |
| | | 3 File(s) | 2,699,718 bytes |
| | | 2 Dir(s) | 14,247,424 bytes free |

```
F:\>_
```

Chapter 4: File system management

4. FAT system

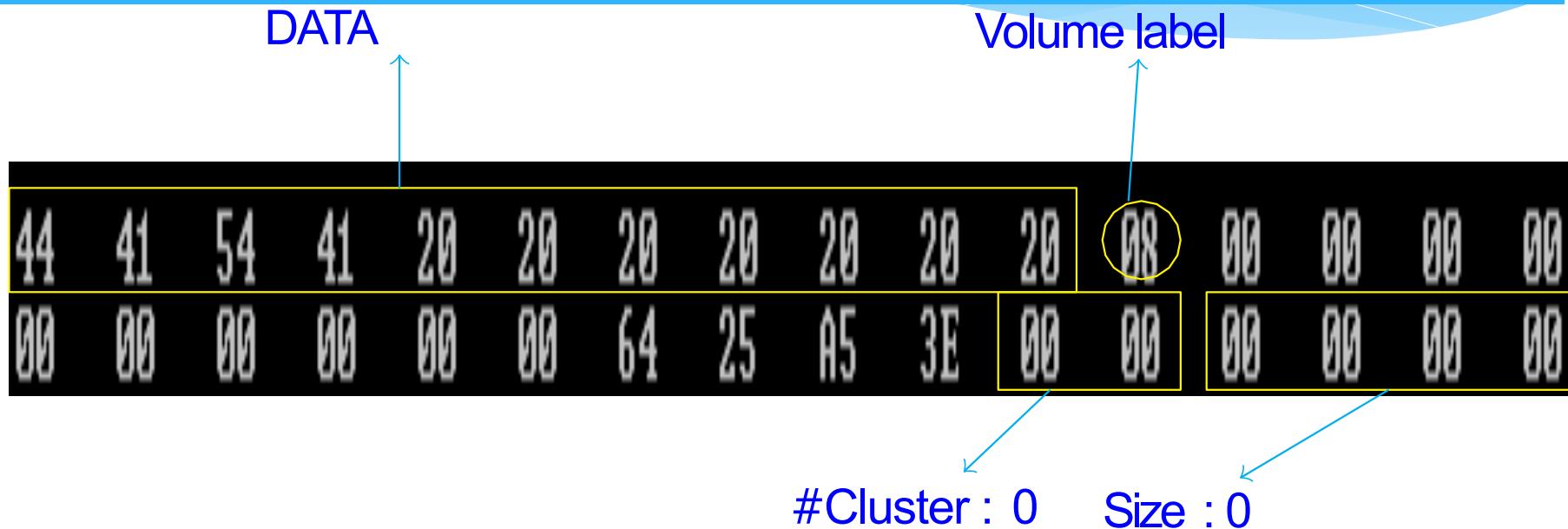
4.3 Root directory

Decode ROOT

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 44 | 41 | 54 | 41 | 20 | 20 | 20 | 20 | 20 | 20 | 08 | 00 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 64 | 25 | A5 | 3E | 00 | 00 | 00 | 00 | 00 | 00 |

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| E5 | 44 | 48 | 20 | 20 | 20 | 20 | 20 | 50 | 44 | 46 | 20 | 18 | 0A | 93 | 34 |
| A5 | 3E | A5 | 3E | 00 | 00 | 6F | 34 | A5 | 3E | 03 | 00 | 38 | 25 | 29 | 00 |

Decode ROOT 1



E5 44 48 20 20 20 20 50 44 46 20 18 0A 93 34

A5 3E A5 3E 00 00 6F 34 A5 3E 03 00 38 25 29 00

ROOT decoding

| DATA | | | | | | | | | | | | Volumne label | | | |
|------|----|----|----|----|----|----|----|----|----|----|----|---------------|----|----|----|
| 44 | 41 | 54 | 41 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 20 | 00 | 00 | 00 | 00 |
| 00 | 00 | 00 | 00 | 00 | 00 | 64 | 25 | A5 | 3E | 00 | 00 | 00 | 00 | 00 | 00 |

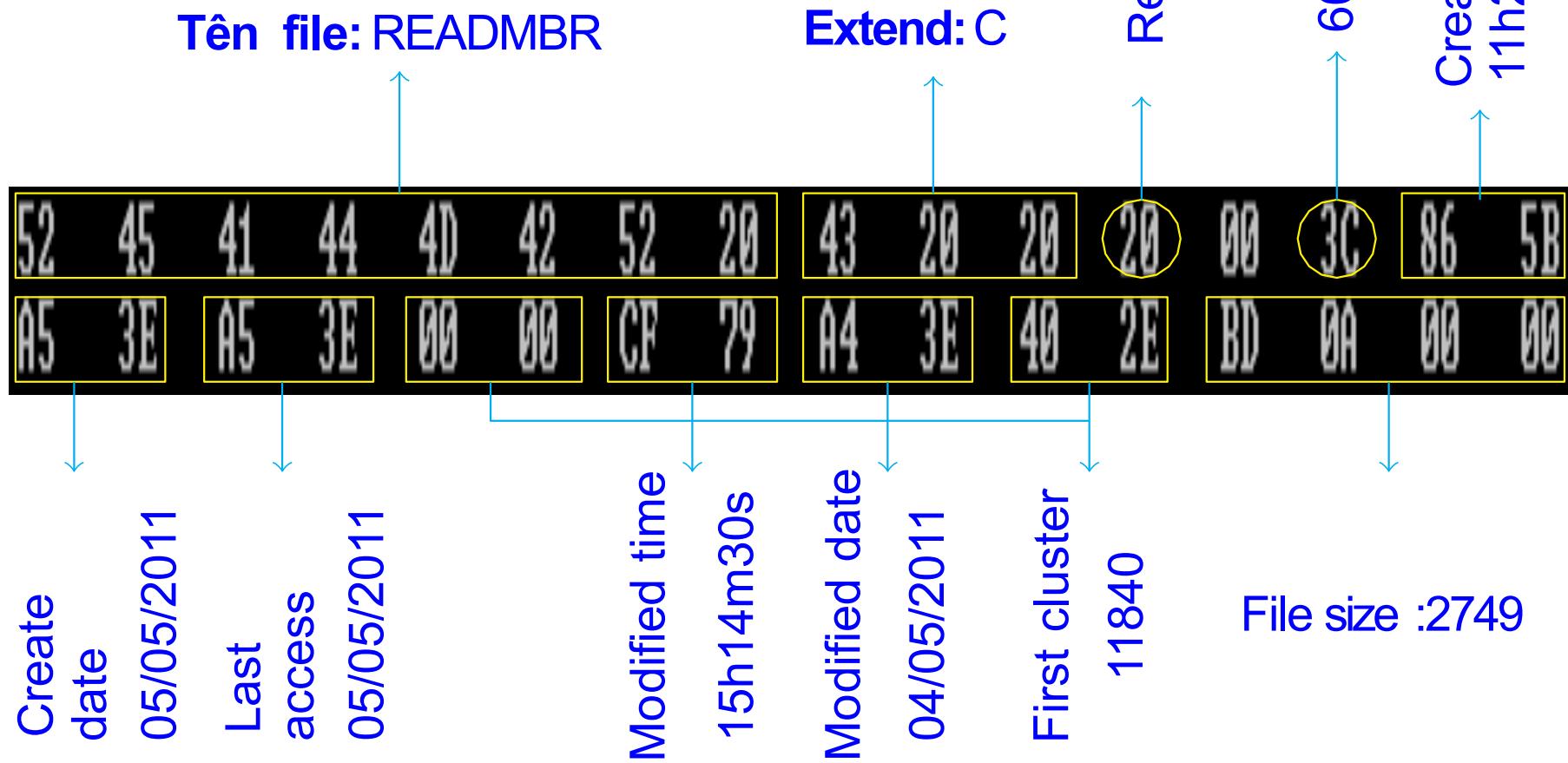
#Cluster : 0 Size : 0

File is removed

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| E5 | 44 | 48 | 20 | 20 | 20 | 20 | 50 | 44 | 46 | 20 | 18 | 0A | 93 | 34 | |
| A5 | 3E | A5 | 3E | 00 | 00 | 6F | 34 | A5 | 3E | 03 | 00 | 38 | 25 | 29 | 00 |

ROOT decoding

File ReadMBR.C



| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 42 | 72 | 00 | 2E | 00 | 63 | 00 | 00 | 00 | FF | FF | 0F | 00 | 43 | FF | FF |
| FF | 00 | 00 | FF | FF | FF | FF |
| 01 | 52 | 00 | 65 | 00 | 61 | 00 | 64 | 00 | 42 | 00 | 0F | 00 | 43 | 69 | 00 |
| 6F | 00 | 73 | 00 | 53 | 00 | 65 | 00 | 63 | 00 | 00 | 00 | 74 | 00 | 6F | 00 |
| 52 | 45 | 41 | 44 | 42 | 49 | 7E | 31 | 43 | 20 | 20 | 20 | 00 | A6 | B2 | 4B |
| A5 | 3E | A5 | 3E | 00 | 00 | 76 | 5C | 9A | 3E | 3F | 2E | D1 | 01 | 00 | 00 |

ROOT decoding

File: **ReadBiosSector.c**

| | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 42 | 72 | 00 | 2E | 00 | 63 | 00 | 00 | 00 | FF | FF | FF | 0F | 00 | 43 | FF | FF |
| FF | 00 | 00 | FF | FF | FF |
| 01 | 52 | 00 | 65 | 00 | 61 | 00 | 64 | 00 | 42 | 00 | 00 | 0F | 00 | 43 | 69 | 00 |
| 6F | 00 | 73 | 00 | 53 | 00 | 65 | 00 | 63 | 00 | 00 | 00 | 00 | 74 | 00 | 6F | 00 |
| 52 | 45 | 41 | 44 | 42 | 49 | 7E | 31 | 43 | 20 | 20 | 20 | 20 | 00 | A6 | B2 | 4B |
| A5 | 3E | A5 | 3E | 00 | 00 | 76 | 5C | 9A | 3E | 3F | 2E | 2E | D1 | 01 | 00 | 00 |

READBI~1.C

Normal 8.3 element

Conclusion