Chapter 2: File System

Operating system

TS. DO TIEN DUNG

dungdt@ptit.edu.vn

Posts and Telecommunications Institute of Technology
Faculty of Information Technology 1

Faculty of Information Technology 1
Posts and Telecommunication Institute of Technology

August 15, 2023

Contents

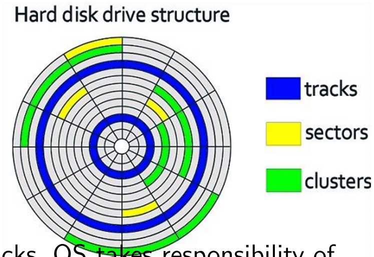
Chapter 2 content

File space allocation

The important task of OS is to manage file system

- Allocate the disk space and external memory to store files and folder.
- Store the allocated space to get access if needed.

  Sector is smallest unit to read or write data
  1 sector = 512byte
  Cluster is allocated unit including some sectors smallest unit that OS allocate for file 2KB-32KB



- Each file includes a set of blocks, OS takes responsibility of allocating blocks for files.
  - Disk space need to allocate for file
  - Mangage the empty space for future allocation

File space allocation (cont.)

- Optimize the space and the access speed on disk.

- Some problems:
  - What is the maximum allocated space at one time for a file?
  - Allocated space for a file is partion, what is the size of partion?

File space allocation (cont.)

Purpose of space allocation for file

- Increase the efficiency of serial access
- Easy to access files randomely
- Easy to manage files

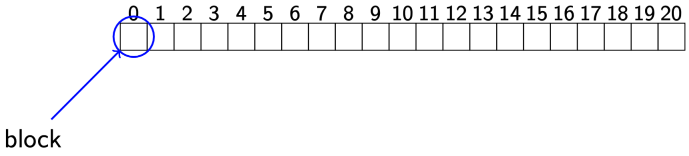
Methods of space allocation for file

- Serial block allocation
- Use the connection list
- Use the connection list on index table
- Use indexed blocks

Serial blocks allocation

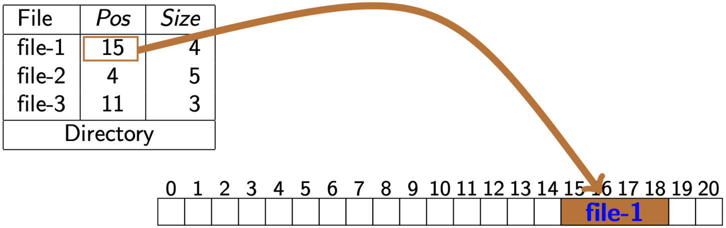
- **Principle:** *file is allocated the blocks in series*
- OS chooses one space which has enough blocks for file
- Allocation Table for file location identification, including: file title, block start, number of blocks (length) of file
- OS knows the file size at creating time

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



Serial blocks allocation (cont.)

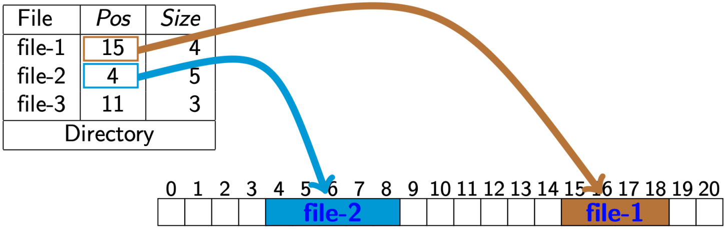
- **Principle:** *file is allocated the blocks in series*
- OS chooses one space which has enough blocks for file
- Allocation Table for file location identification, including: file title, block start, number of blocks (length) of file
- OS knows the file size at creating time



https://www.

Serial blocks allocation (cont.)

- **Principle:** *file is allocated the blocks in series*
- OS chooses one space which has enough blocks for file
- Allocation Table for file location identification, including: file title, block start, number of blocks (length) of file
- OS knows the file size at creating time

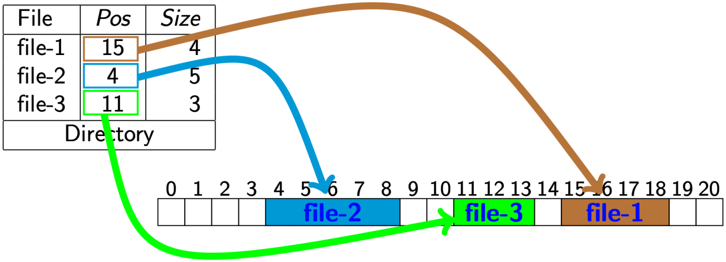


File	*Pos*	*Size*
file-1	15	4
file-2	4	5
file-3	11	3
Directory		

Serial blocks allocation (cont.)

Serial blocks allocation

- **Principle:** *file is allocated the blocks in series*
- OS chooses one space which has enough blocks for file
- Allocation Table for file location identification, including: file title, block start, number of blocks (length) of file
- OS know the file size at file creating time



Serial blocks allocation (cont.)

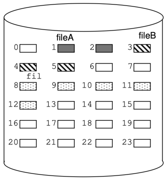
Serial blocks allocation

Advantages:

+ Allow accessing directly and serially
+ Simple, save time of reading blocks

Disadvantages:

- Need to know the file volume in advance
- Be difficult to find enough space on disk
- OS needs to check the empty space
- Create fragmentation



Thư mục

Tên file	Bắt đầu	Độ dài
fileA	1	2
fileB	3	3
fileB	8	5

** Fragmentation: the phenomenon that the remaining free space on the disk is too small, so it cannot be allocated to a larger file*

Space allocation for file
Allocation by connection list

- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.

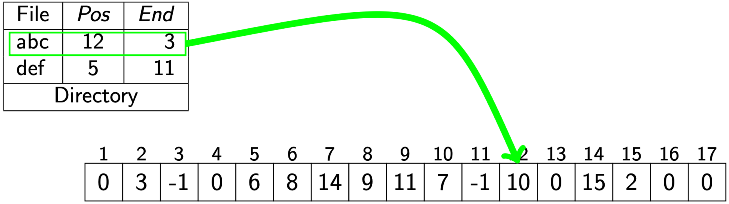
File	*Pos*	*End*
abc	12	3
def	5	11
Directory		

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	3	-1	0	6	8	14	9	11	7	-1	10	0	15	2	0	0

Space allocation for file (cont.)

Allocation by connection list

- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.



File	*Pos*	*End*
abc	12	3
def	5	11
Directory		

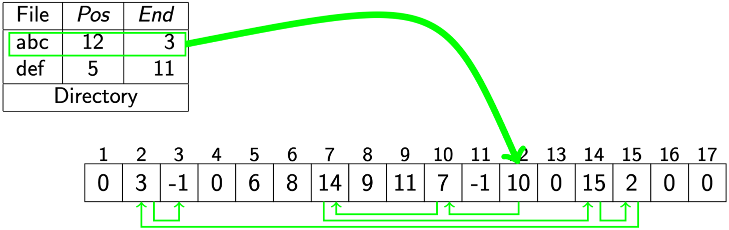
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	3	-1	0	6	8	14	9	11	7	-1	10	0	15	2	0	0

Space allocation for file (cont.)
Allocation by connection list

- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.

File	*Pos*	*End*
abc	12	3
def	5	11

Directory

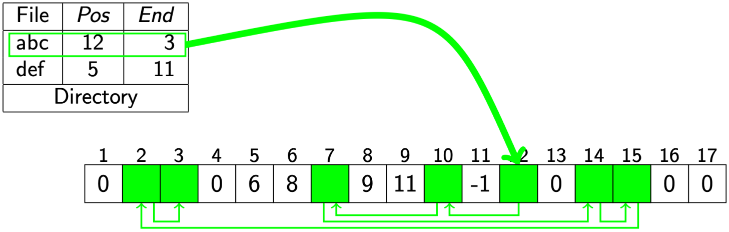


1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	3	-1	0	6	8	14	9	11	7	-1	10	0	15	2	0	0

Space allocation for file (cont.)

Allocation by connection list

- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.



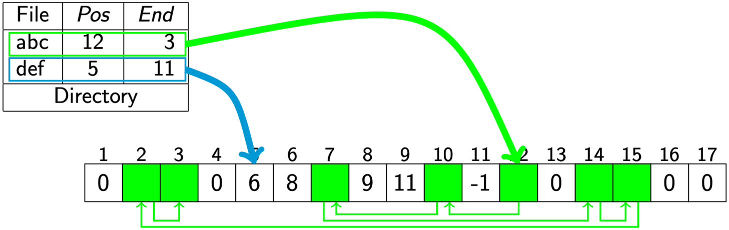
File	*Pos*	*End*
abc	12	3
def	5	11
Directory		

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	🟩	🟩	0	6	8	🟩	9	11	🟩	-1	🟩	0	🟩	🟩	0	0

File abc gồm 7 khối: 12, 10, 7, 14, 15, 2, 3

Space allocation for file (cont.)
Allocation by connection list

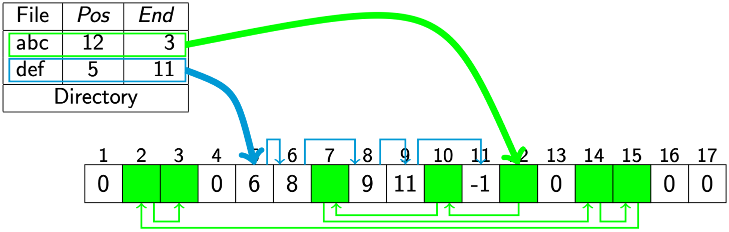
- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.



File abc gồm 7 khối: 12, 10, 7, 14, 15, 2, 3

Space allocation for file (cont.)
Allocation by connection list

- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.



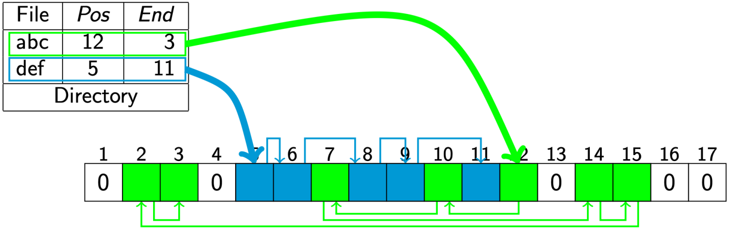
File	*Pos*	*End*
abc	12	3
def	5	11
Directory		

File abc gồm 7 khối: 12, 10, 7, 14, 15, 2, 3

Space allocation for file (cont.)

Allocation by connection list

- **Principle:** *File is allocated blocks connected to each other to create connection list, the beginner of block has pointer pointing the next block.*
- File is allocated a new block. That block is added at the end of list.
- Allocation table has pointer of first block. OS reads each block and use pointer to identify the next block.



File abc has 7 blocks: 12, 10, 7, 14, 15, 2, 3

File def has 5 blocks: 5, 6, 8, 9, 11

Space allocation for file (cont.)

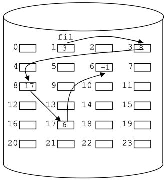
Allocation by connection list

Advantages:

+ No defragment since blocks are not necessary to be next to each other
+ No need to know file volume in advance
+ Easy to find file location, simple indexes

Disadvatanges:

- Do not support direct access
- To read a block, it needs to read from the first block to the needed block
- When the value of pointer changes, the identification of blocks equivalent file is not correct
- Access speed is not fast

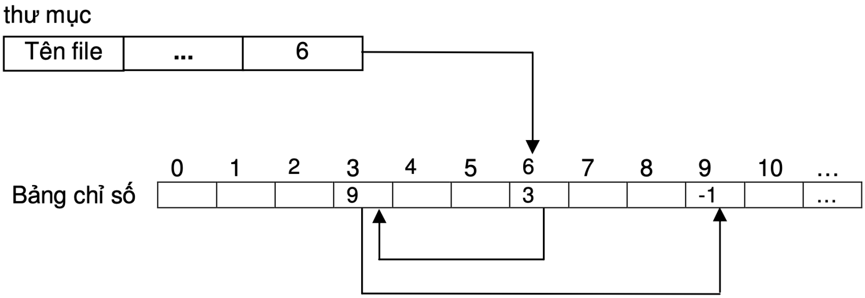


Thư mục

Tên file	Bắt đầu
fileA	1

Space allocation for file
Allocate by connection list based on index

- Index table: each cell is equivalent to one sector of disk
- Pointer to next block of file is stored in equivalent cell of table.
- Each logic disk has one index table stored in a identified location.
- Size of table depends on the block number on disk.



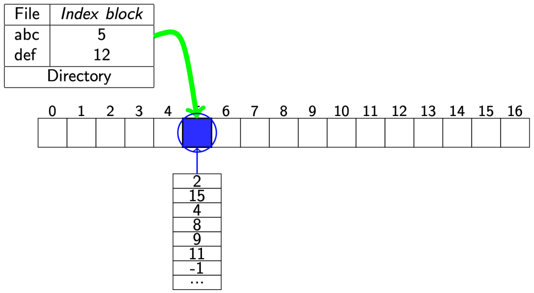
Space allocation for file (cont.)

Allocate by connection list based on index

- Allow accessing to file directly: follow series of pointer in index table
- FAT table (File Allocation Table) is stored at the head of logic disk after start sector được lưu ở đầu mỗi đĩa logic sau sector khởi động
- FAT12, FAT16, FAT32: Each ceall of table has size of 12, 16, 32 bit allowing to manage maximum $2^{12}, 2^{16}, 2^{32}$

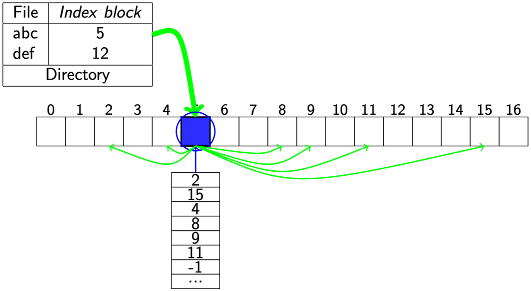
Space allocation for file
Allocation by index block

- **Principle:** *Each file has its own index block including list of file indexes*
- All pointers pointing to blocks of one file is concentrated at one place, called I-node
- Array has file properties and locations of blocks on the disk.
- Cell number i of array has pointer pointing to block number i of file.



Space allocation for file (cont.)
Allocation by index block

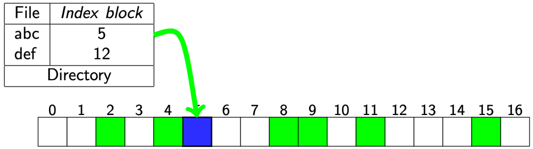
- **Principle:** *Each file has its own index block including list of file indexes*
- All pointers pointing to blocks of one file is concentrated at one place, called I-node
- Array has file properties and locations of blocks on the disk.
- Cell number i of array has pointer pointing to block number i of file.


File	*Index block*
abc	5
def	12

Directory

Space allocation for file (cont.)
Allocation by index block

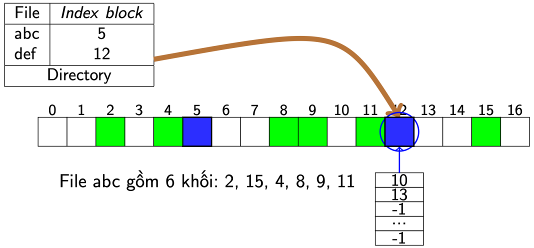
- **Principle:** *Each file has its own index block including list of file indexes*
- All pointers pointing to blocks of one file is concentrated at one place, called I-node
- Array has file properties and locations of blocks on the disk.
- Cell number i of array has pointer pointing to block number i of file.



File abc gồm 6 khối: 2, 15, 4, 8, 9, 11

Space allocation for file (cont.)
Allocation by index block

- **Principle:** *Each file has its own index block including list of file indexes*
- All pointers pointing to blocks of one file is concentrated at one place, called I-node
- Array has file properties and locations of blocks on the disk.
- Cell number i of array has pointer pointing to block number i of file.


File	*Index block*
abc	5
def	12

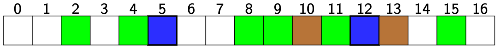
Directory

File abc gồm 6 khối: 2, 15, 4, 8, 9, 11

Space allocation for file (cont.)
Allocation by index block

- **Principle:** *Each file has its own index block including list of file indexes*
- All pointers pointing to blocks of one file is concentrated at one place, called I-node
- Array has file properties and locations of blocks on the disk.
- Cell number i of array has pointer pointing to block number i of file.

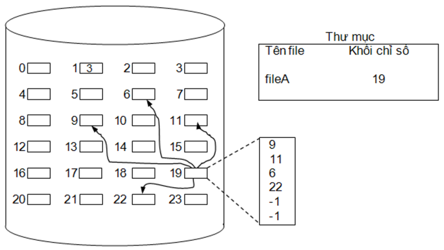
File	*Index block*
abc	5
def	12
Directory	



File abc gồm 6 khối: 2, 15, 4, 8, 9, 11
File def gồm 2 khối: 10, 13

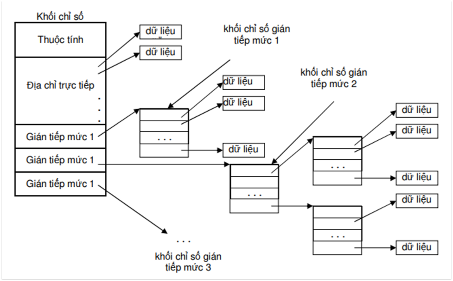
Space allocation for file (cont.)
Allocation by index block

- Choose size of I-node:
  - Small: save the space but do not have enough pointer for big file.
  - Big: wasted for small file.
- Solution:
  - Change size of i-node = use connection list
  - Use many levels I-node structure



Space allocation for file (cont.)
Allocation by index block

- Advantage:
  - Allow access directly
  - The blocks of one file is not next to each other continuously.
- Disadvantage:
  - Slow file access



Manage empty space on disk

- Space on disk is allocated by blocks without dividing management. To manage space effectively, in addition to the block allocation method, it needs to be caution:
  - Choose the size of block
  - Manage the empty block
- Block size:
  - Block is the smallest allocated unit of OS, each file include the interger number of blocks.

Manage empty space on disk (cont.)

- Big block size:
  - Decrease the size of index table, speed up the file reading spead
  - Inner fragmentation
- Small block size:
  - One file has many blocks, distributed descretedly.
  - Slow file reading time.
- Block size depends on:
  - Disk size: big disk, choose big block size => time access is fast, simplify management.
  - File size: system has many big files, block size inrcrease and vice versa.
- The block size is usually a power of 2 of the sector and ranges from 512B to 32 KB

Manage empty space on disk (cont.)

- To be able to allocate memory blocks on disk for files, OS needs to know which blocks are currently free. Empty blocks include blocks that have never been allocated or were allocated but have been freed.
- Free space management methods:
  - Bit table
  - Linked list
  - Empty spaces list

Manage free space on disk
Bit table

- Bit vector is a 1-dimensional array
- Each cell is 1 bit in size and corresponds to a block on the disk
- The allocated block: 0, empty block: 1 or vice versa
- Easy to find a group of consecutive empty blocks
- With large disks, reading the entire bit vector into memory may require much memory space

For instance: In the following case, blocks 0, 1, 8, 9, 10 have been allocated, the remaining blocks are unused:

0011111100011111...

Manage free space on disk

Danh sách kết nối

- Empty blocks are linked together into lists
- Each empty block contains a pointer to the next empty block
- The address of the first free block is stored in a special location on disk and is kept by OS to work with files when needed.
- Requires sequential access to browse this list
- The OS can immediately allocate blocks at the beginning of the list.

Manage free space on disk
Empty spaces list

- Adjacent blocks are often allocated and released at the same time
- Stores the position of the first free block of consecutive free blocks and the number of following free blocks.
- The above information is saved in a separate list.

File system reliability

- File is a place to store information that is very important to users, and to the computing system itself.
- The file system must have high sustainability, high storage security to make sure that information is intact and no lost.
- File system information stored on disks and other external memory devices may be corrupted or lost due to:
  - Hardware/software error
  - Technical error
  - Although it is impossible to prevent such incidents => it still needs to be built so that it is possible to detect and fix the consequences at highest ability.

File system reliability
Detect and eliminate bad blocks

- During production and use, the disk may contain damaged blocks and cannot be used to read/write information.
  - Floppy disks and damaged blocks often appear during use
  - Hard disk: often has damaged blocks during the manufacturing process. These bad blocks are detected by the manufacturer during disk testing and are stored in the bad block list that comes with the disk.

File system reliability (cont.)
Detect and eliminate bad blocks

- Detect and eliminate bad blocks
  - Method 1:
    - A sector on the disk is reserved to contain a list of bad blocks
    - Some non-bad blocks are reserved for reservation
    - Bad blocks are replaced by reserved blocks using address replacement
    - Access to failed block convert to access to reserved block
  - Method 2:
    - Concentrate all broken blocks into one file
    - Allocated and no longer used

File system reliability
Backup

- Creates a copy of the disk on another medium
- Full backup:
  - Burn all information on the disc to another media
  - Consistant but it takes a lot of time
- Tncremental backup:
  - Used after performing a full backup at least once
  - Only save files that have been changed after the last backup
  - The system stores information about file storage times
  - DOS: file changes, archive bit =1
- Combine:
  - Full backup: weekly/monthly
  - Incremental backup: everyday

File system reliability
Check the integrity of the file system

- The file system contains many database:
  - If link information is damaged, the integrity of the system is broken
  - The link is broken, do not know which is next block
- The blocks are not in the list of empty blocks, nor in any file
- A block can both belong to a certain file and is in the empty block list
- OS has programs that check the integrity of the file system, which are run when the system boots, especially after a crash.

For instance: SCANDISK in WIN or DOS

File system reliability
Ensure integrity using transactions

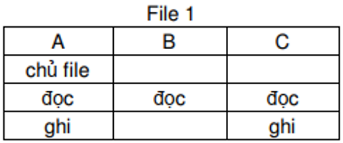
- Although it is possible to check the integrity of links in the file system. However, checking the entire file system requires a lot of time. Testing only allows detecting errors after they have occurred and does not guarantee data recovery for some errors.
- A transaction is a set of operations that need to be performed completely together With file systems: each transaction will include change operations that need to be performed together (For example, link update operations related to a block on disk)
- The entire file system status is recorded in the log file
- Every time a transaction is performed, the system checks to see if the transaction has been completed completely.
- If the transaction is not fully committed, the OS uses information from the log to restore the file system to an error-free state before committing the transaction.

File system security

- Prevent unauthorized access to information stored in files and folders
- For small systems for one user, the security issue is relatively simple, it can be done by physical measures => Do not allow others to access the system,
- In multi-user computing systems, security of files and directories is done by:
  - Control access operations to files or folders
  - Using password => Users must remember many passwords => Every time they operate with resources, they must type the password
- Use Access Control List
  - Each file is assigned a list, containing information identifying the user and the rights that person has with the file
  - ACLs are often stored as file/folder attributes
  - Often used in conjunction with a login mechanism

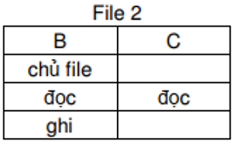
File system security (cont.)

- Basic access rights:
  - Read permission (r): people with this permission are allowed to read the file content
  - Right to write and change (w): allowed to write to the file, that is, change the file content
  - Right to delete: allowed to delete files, this right is equivalent to the right to change files
  - Change owner
- Use ACL (Access Control List)



File 1

A	B	C
chủ file		
đọc	đọc	đọc
ghi		ghi



File 2

B	C
chủ file	
đọc	đọc
ghi	

FAT file system

- The FAT system was designed for use in DOS, and was later used in several versions of Windows 3.0,3.1,95/98.
- Currently, FAT is still a popular system, used in most OSs today to manage memory cards, floppy disks, and CDs.
- 3 versions: FAT12, FAT16, FAT32
- Numbers are FAT table size: 12, 16 và 32 bit
- Currently, FAT32 is often used for hard disks, FAT16 is used for small capacity external memory devices such as CDs and external memory cards.

Tổng kết

Chương 2

- Cấp phát không gian cho file
- Quản lý không gian trống trên đĩa
- Độ tin cậy của hệ thống file
- Bảo mật cho hệ thống file

Tiếp theo

- Hệ thống file FAT
  - Đĩa logic
  - Boot sector
  - Bảng FAT
  - Root Thư mục gốc
  - Hàm đọc đĩa
  - Bài tập thực hành