953214
Operating System and Computer Network

Chapter 7

File Management

Lecturer: Dr. Phudinan Singkhamfu, Dr. Parinya Suwansrikham

CAMT
College of Arts, Media and Technology
Chiang Mai University

File Management

- One of the important services of OS
- Objectives
  - Meet the data requirements of the user
  - Guarantee valid data
  - Optimize performance -Both throughput and response time
  - Support a wide variety of devices
  - Minimize lost or destroyed data
  - Provide a standard set of I/O routines
  - Provide support for multiple users

OS and File Operations

- System calls to (for example) create files, delete files, move files, rename files, copy files, open files, close file, read files, write files.
- Open (*Fi*) – search the directory structure on disk for entry *Fi*, and move the content of entry to memory.
- Close (*Fi*) – move the content of entry *Fi* in memory to directory structure on disk.
- Control other's access to files
- Able to move data between files
- Back up and recover files

File Terms

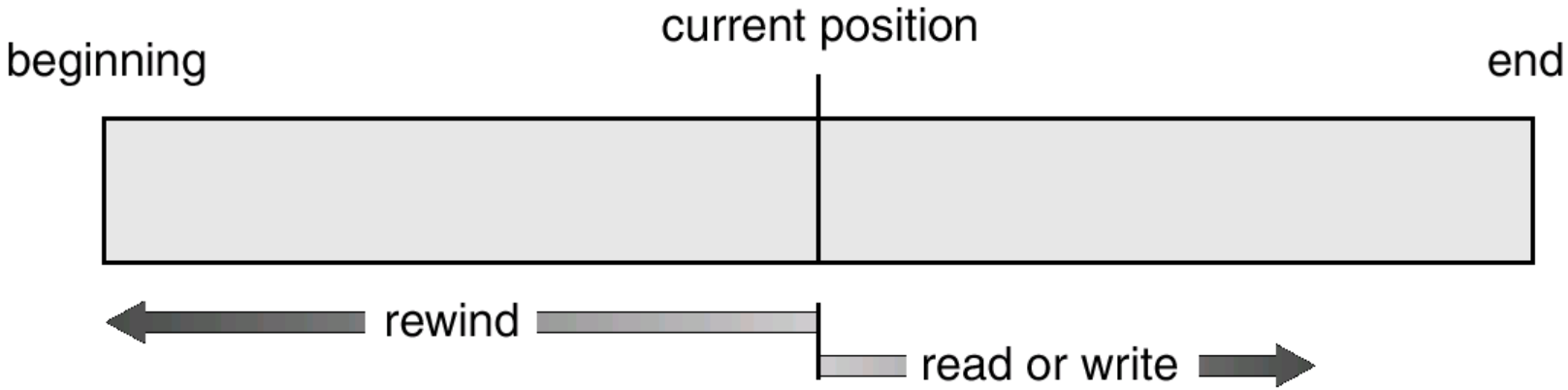
Field	Field is the basic element of data, such as an employee's last name, a date, or the value of a sensor reading.
Record	Record is a collection of related fields that can be a unit by some application program. For example, an employee record contain such fields as name, social security number, and so on.
File	File is a collection of similar records. The file is treated as a single entity by users and applications and may be referenced by name.
Database	Data base is a collection of related data. The essential aspects of a database are that the relationships that exist among elements of data are explicit,

Access Methods

- It provides a standard interface between applications and the file systems and devices that hold the data.

Access Methods

- Sequential Access
  - *read next*
  - *write next*
  - *reset*
  - No read after last write
  - *(rewrite)*
- Direct Access
  - *read n*
  - *write n*
  - *position to n*
  - *read next*
  - *write next*
  - *rewrite n*
- *n= relative block number*

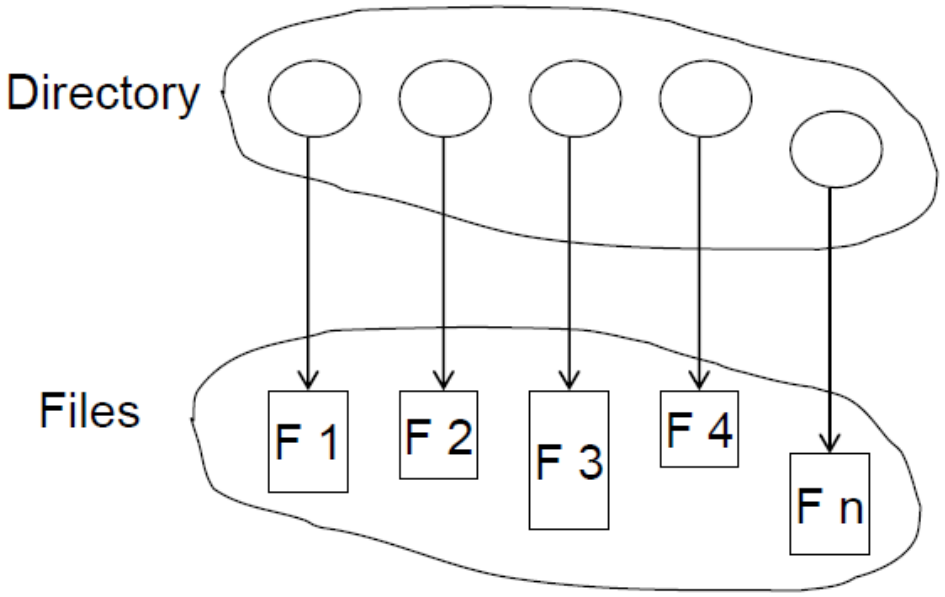


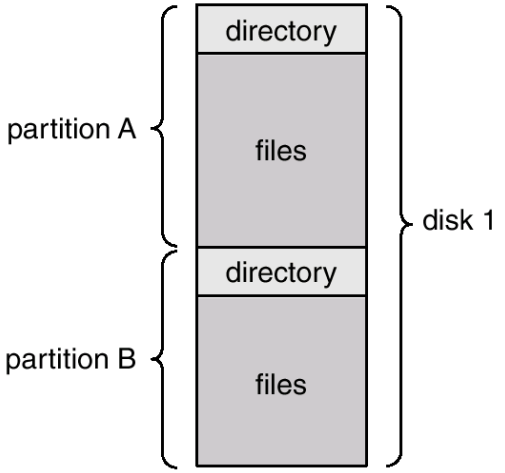
File Directory

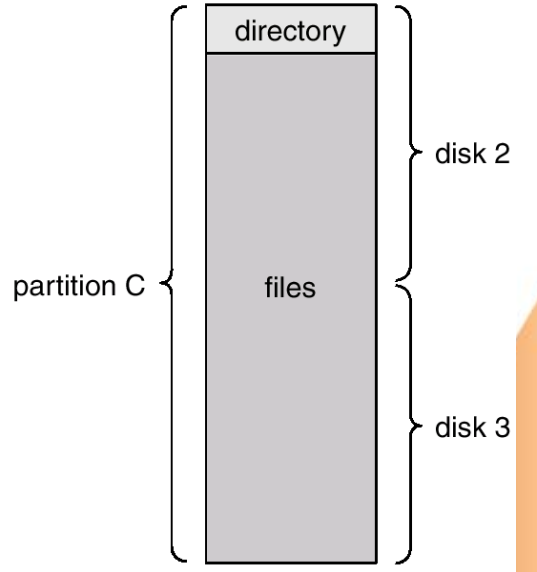
- Associated with any file management system and collection of files is a file directory.
- Basic information of file directory,
  - File name: must be unique within a specific directory.
  - File type: text, binary, executable, etc. The file type can be specified by file extension, .jpg, .png, .docx, etc.
  - File organization: For systems that support different organizations

Directory Structure

- A collection of nodes containing information about all files.







Both the directory structure and the files reside on disk.
Backups of these two structures are kept on tapes.

Directory Implementation

- Linear list of file names with pointer to the data blocks.
  - simple to program
  - time-consuming to execute
- Hash Table –linear list with hash data structure.
  - decreases directory search time
  - *collisions*–situations where two file names hash to the same location
  - fixed size

File Sharing

- Sharing of files on multi-user systems is desirable.
- Sharing may be done through a *protection* scheme.
- On distributed systems, files may be shared across a network.
- Network File System (NFS) is a common distributed file-sharing method.

Protection

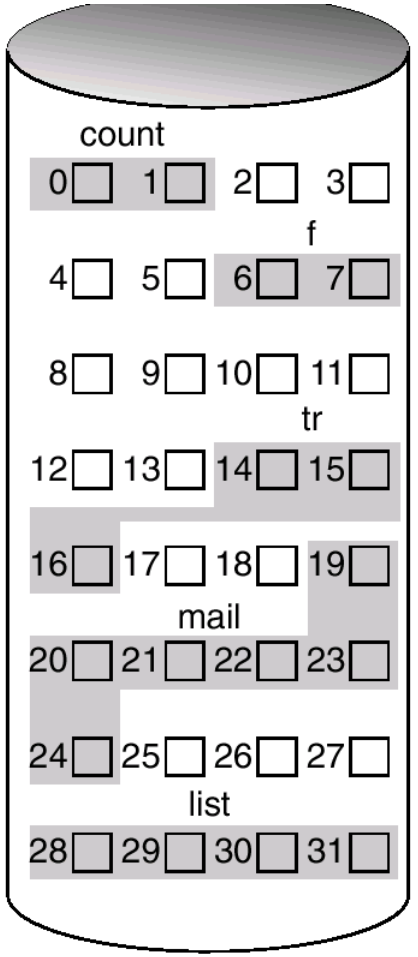
- File owner/creator should be able to control:
  - what can be done
  - by whom
- Types of access
  - Read
  - Write
  - Execute
  - Append
  - Delete
  - List

Allocation Methods

- An allocation method refers to how disk blocks are allocated for files
- Three main methods:
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation

Contiguous Allocation

- Each file occupies a set of contiguous blocks on the disk.
- Simple –only starting location (block #) and length (number of blocks) are required.
- Random access.
- Wasteful of space (dynamic storage-allocation problem).
- Files cannot grow.

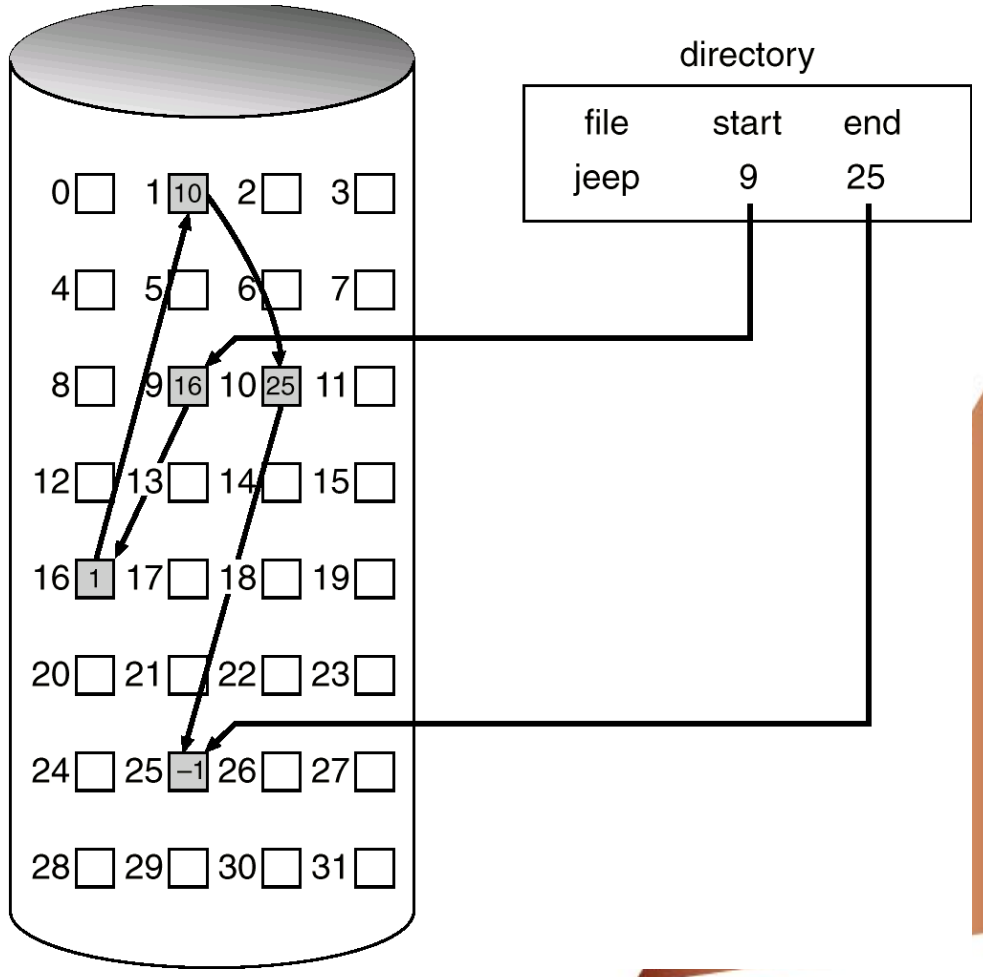


directory

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

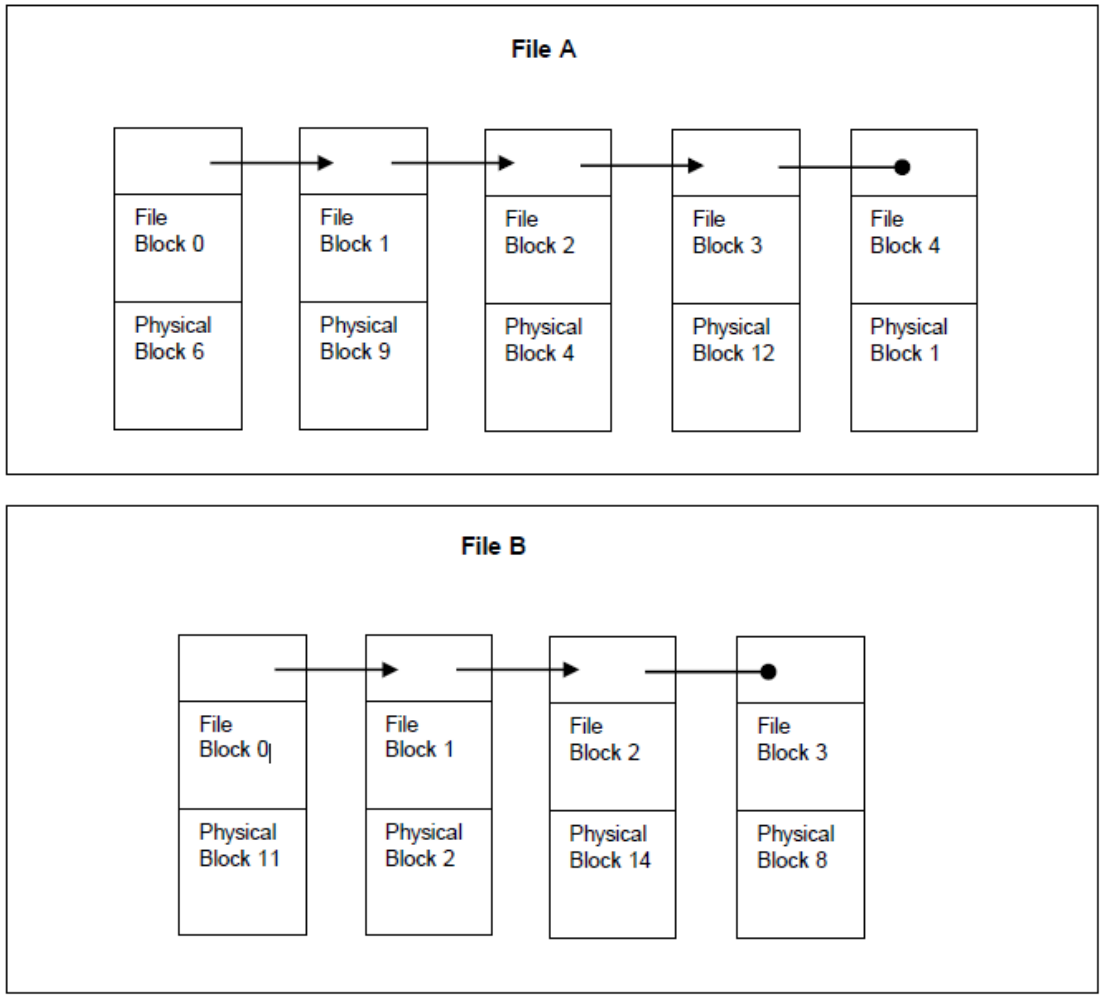
Linked Allocation

- Each file is a linked list of disk blocks: blocks may be scattered anywhere on the disk.
- Simple –need only starting address
- Free-space management system –no waste of space
- No random access
- Two file system implementation methods (mapping)that uses linked lists:
  - Linked chain of blocks
  - File-allocation table (FAT)



File implementation -Linked chain

- Block to be accessed is the Qth block in the linked chain of blocks representing the file.
- Displacement into block = R + 1



Linked chain -advantages

- Every block can be used, unlike a scheme that insists that every file is contiguous.
- No space is lost due to external fragmentation (although there is fragmentation within the file, which can lead to performance issues).
- The directory entry only has to store the first block. The rest of the file can be found from there.
- The size of the file does not have to be known beforehand (unlike a contiguous file allocation scheme).
- When more space is required for a file any block can be allocated (e.g. the first block on the free block list).

Linked chain -disadvantages

- No Random access is very slow (as it needs many disc reads to access a random point in the file).
  - The implementation is really only useful for sequential access.
- Space is lost within each block due to the pointer.
- The number of bytes is not a power of two.
  - This is not fatal but does have an impact on performance.
- Reliability could be a problem.
  - one corrupt block pointer and the whole system might become corrupted (e.g., writing over a block that belongs to another file).

File implementation –FAT

- File-Allocation Table (FAT)
- This method removes the pointers from the data block and places them in a table which is stored in memory.

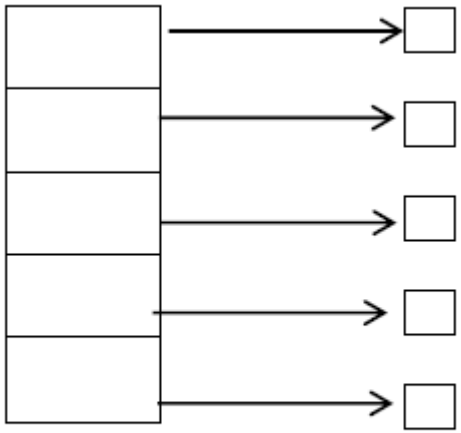
Physical Block	Pointers	
0		← Unused block
1	0	
2	14	
3		
4	12	
5		
6	9	← File A starts here
7		
8	0	
9	4	
10		
11	2	← File B starts here
12	1	
13		
14	8	

FAT -advantages and disadvantages

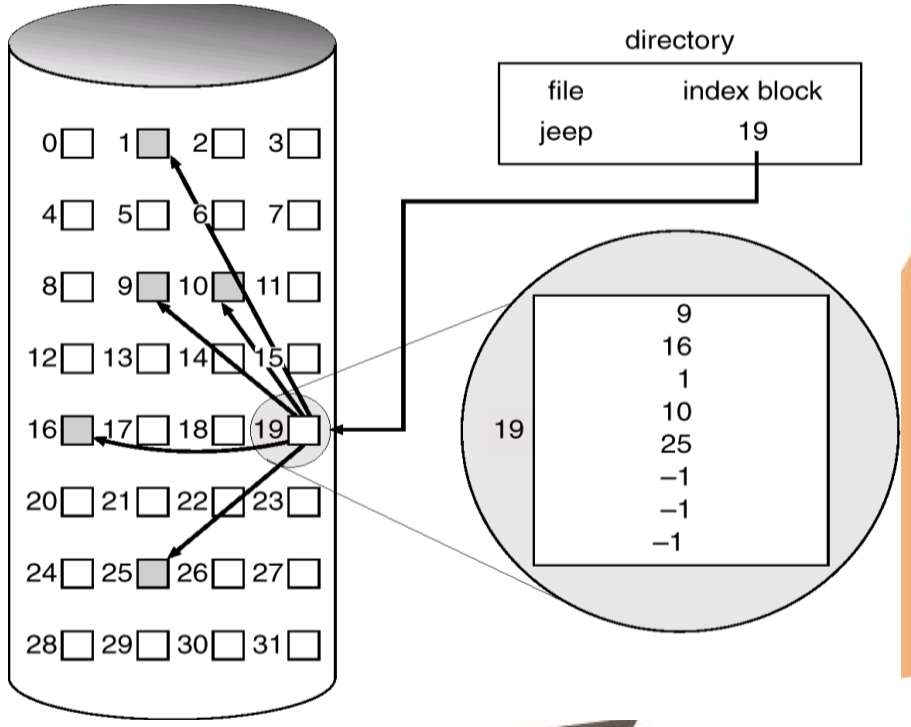
- Advantages
  - The entire block is available for data.
  - Random access can be implemented a lot more efficiently.
  - Although the pointers still have to be followed these are now in main memory and are thus much faster.
- Disadvantages
  - The entire table must be in memory all the time.
    - For a large disc (with a large number of blocks) this can lead to a large table having to be kept in memory.

Indexed Allocation

- Brings all pointers together into the index block.
- Logical view.



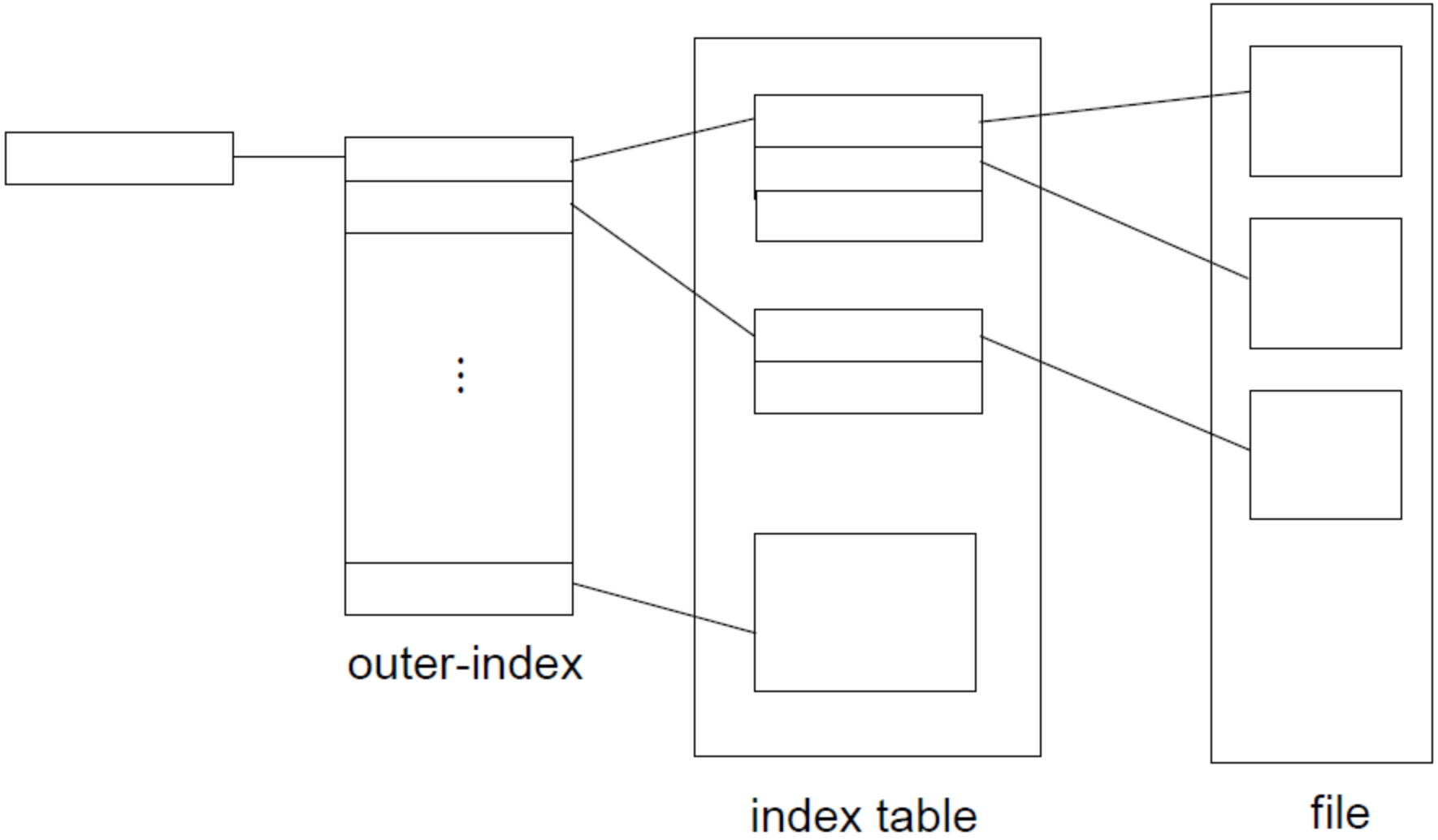
index table



Indexed Allocation (Cont.)

- Need index table
- Random access
- Dynamic access without external fragmentation but have overhead of index block.
- Indexed Allocation –Mapping
  - Mapping from logical to physical in a file of unbounded length (block size of 512 words). We need only 1 block for index table.
  - Linked scheme –Link blocks of index table (no limit on size).
  - Two-level index (maximum file size is 5123)

Indexed Allocation –Mapping (Cont.)



Efficiency and Performance

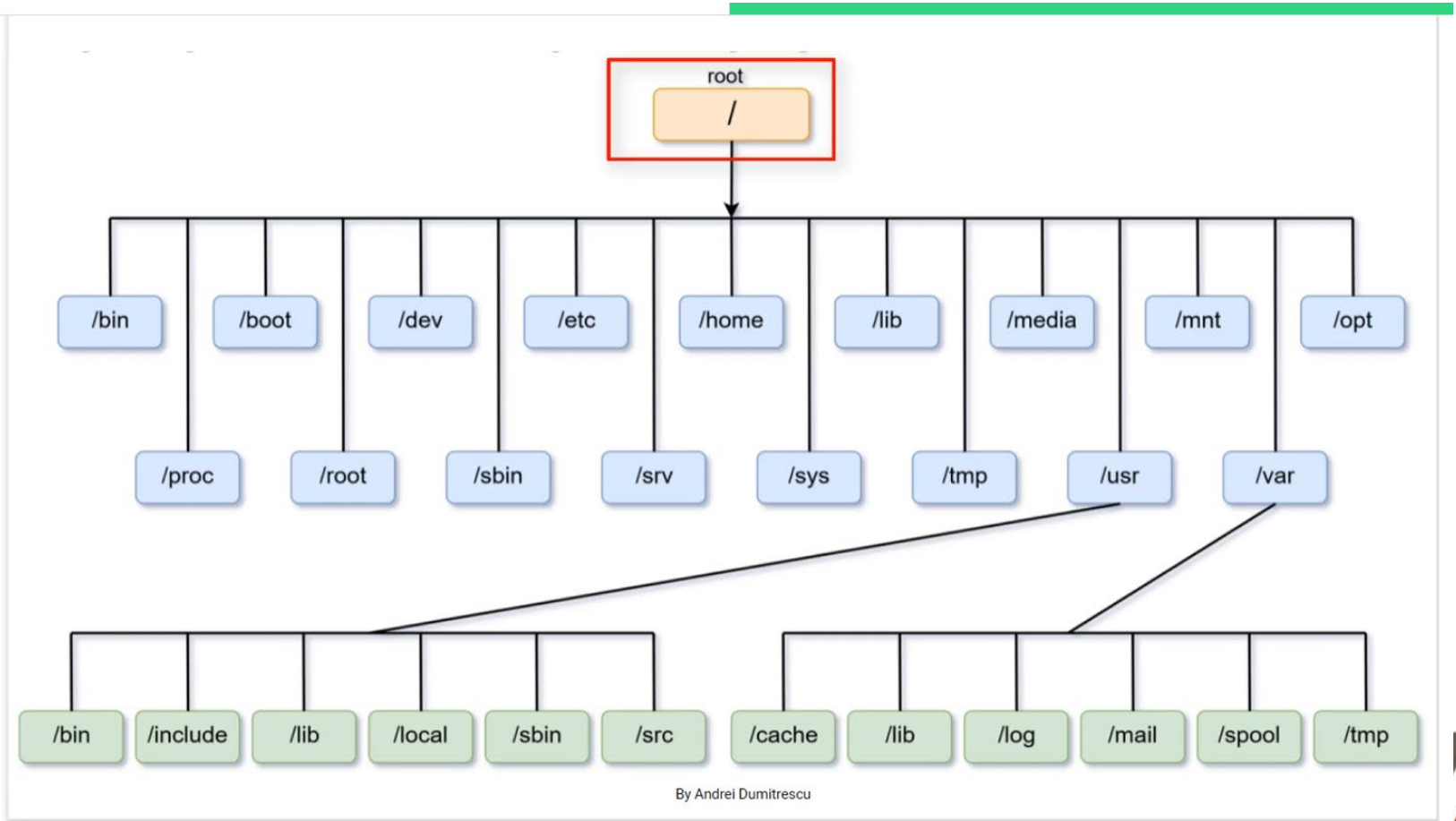
- Efficiency dependent on:
  - disk allocation and directory algorithms
  - types of data kept in file's directory entry
- Performance
  - disk cache –separate section of main memory for frequently used blocks
  - free-behind and read-ahead –techniques to optimize sequential access
  - improve PC performance by dedicating section of memory as virtual disk, or RAM disk.

Summary

- File management is one of the important services of OS
- Three main methods to allocate disk blocks to a file
  - Contiguous allocation
  - Linked allocation
  - Indexed allocation
- Two common file system implementation methods that use linked lists are:
  - Linked chain of blocks
  - File-allocation table (FAT)

- Readings:
  - OSC, Silberschatz, Galvin and Gagne chapter 11

Linux File Structure



Linux File Structure

- Absolute path
  - /home/usr/CAMT/
- Relative path
  - ../etc/host/
- File command
  - cd
  - ls
  - tree -> sudo apt install tree
- File permission
  - chmod
  - u-user, g-group, o-other
  - d-directory, r-read, w-write, x-execute