Dr. Hakim Mellah
Department of Computer Science & Software Engineering
Concordia University, Montreal, Canada

Lecture 13: File System

COMP 346: Operating Systems

These slides has been extracted, modified and updated from original slides of :

- ***Operating System Concepts, 10th Edition***, by: S**ilberschatz/Galvin/Gagne**, published by John Wiley & Sons

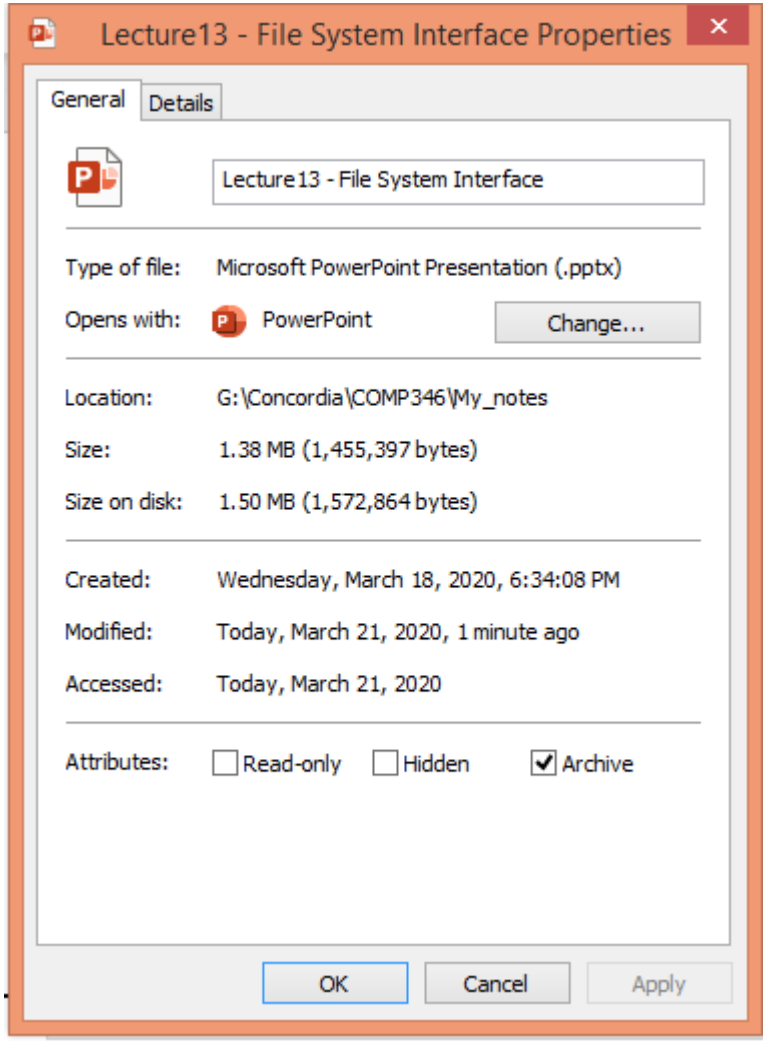
File Concept

- Contiguous logical address space
- Types:
  - Data
    - ✓ numeric
    - ✓ character
    - ✓ binary
  - Program
- Contents defined by file's creator
  - Many types
    - ✓ Consider **text file, source file, executable file**

File Attributes

- **Name** – only information kept in human-readable form
- **Identifier** – unique tag (number) identifies file within file system
- **Type** – needed for systems that support different types
- **Location** – pointer to file location on device
- **Size** – current file size
- **Protection** – controls who can do reading, writing, executing
- **Time, date, and user identification** – data for protection, security, and usage monitoring
- Information about files are kept in the directory structure, which is maintained on the disk

File info Window on Mac OS X and Windows



File Operations

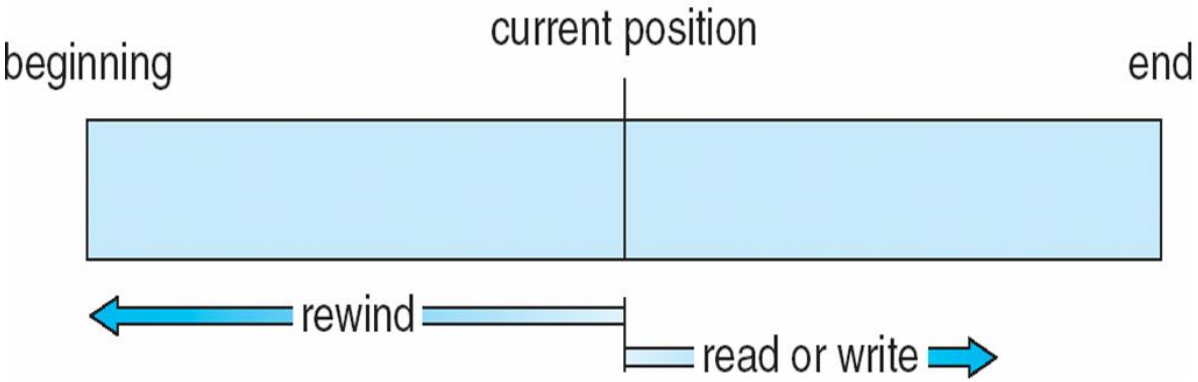
- File is an **abstract data type**
  - **Create**
  - **Write –** at **write pointer** location
  - **Read –** at **read pointer** location
  - **Reposition within file - seek**
  - **Delete**
  - **Truncate**
  - ***Open(F_i)*** – search the directory structure on disk for entry F_i, and move the content of entry to memory
  - ***Close (F_i)*** – move the content of entry F_i in memory to directory structure on disk

File Types – Name, Extension

file type	usual extension	function
executable	exe, com, bin or none	ready-to-run machine-language program
object	obj, o	compiled, machine language, not linked
source code	c, cc, java, pas, asm, a	source code in various languages
batch	bat, sh	commands to the command interpreter
text	txt, doc	textual data, documents
word processor	wp, tex, rtf, doc	various word-processor formats
library	lib, a, so, dll	libraries of routines for programmers
print or view	ps, pdf, jpg	ASCII or binary file in a format for printing or viewing
archive	arc, zip, tar	related files grouped into one file, sometimes compressed, for archiving or storage
multimedia	mpeg, mov, rm, mp3, avi	binary file containing audio or A/V information

Access Methods

- Sequential access
- Direct access
- Other access methods



Allocation Methods

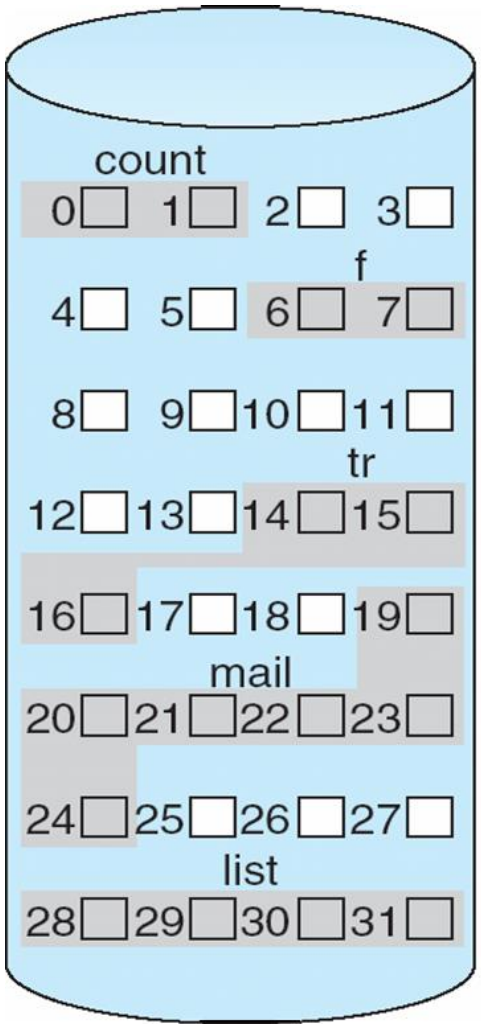
An allocation method refers to how disk blocks are allocated for files:

- **Contiguous allocation**
- **Linked allocation**
- **Indexed allocation**

Contiguous allocation

- Each file occupies set of contiguous blocks
  - Best performance in most cases
  - Simple – only starting location (block #) and length (number of blocks) are required
  - Problems include finding space for file, knowing file size, external fragmentation, need for **compaction off-line** (**downtime**) or **on-line**

Contiguous Allocation



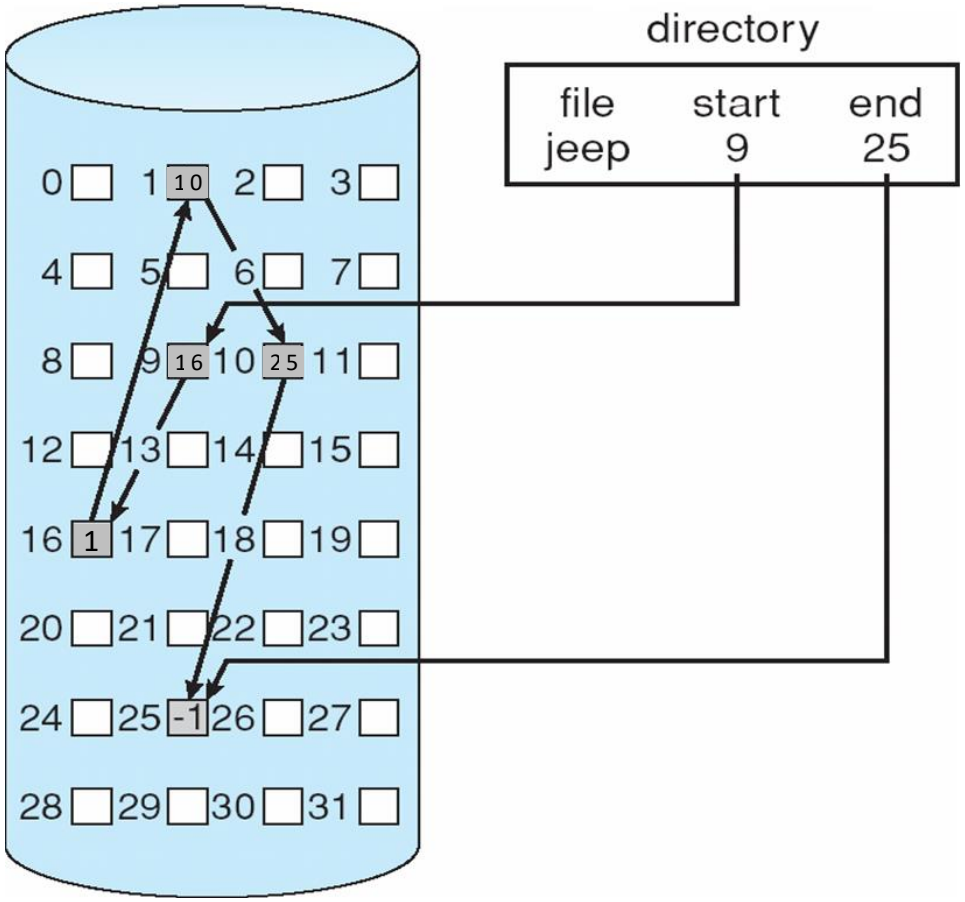
directory

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

Allocation Methods - Linked

- **Linked allocation** – each file a linked list of blocks
  - File ends at nil pointer
  - No external fragmentation
  - Each block contains pointer to next block
  - No compaction, external fragmentation
  - Free space management system called when new block needed
  - Improve efficiency by clustering blocks into groups but increases internal fragmentation
  - Reliability can be a problem
  - Locating a block can take many I/Os and disk seeks

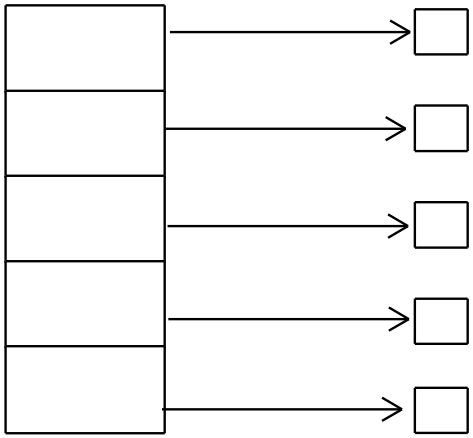
Linked Allocation



Allocation Methods - Indexed

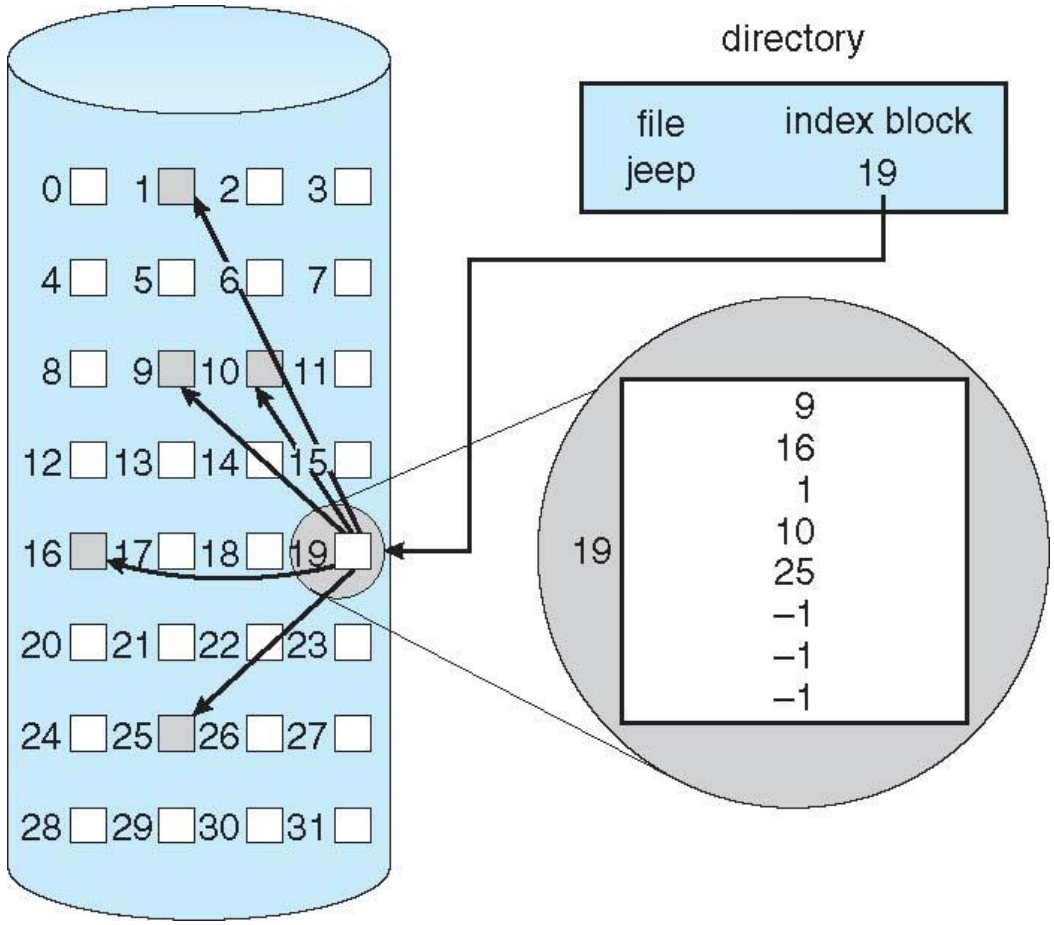
- **Indexed allocation**
  - Each file has its own **index block**(s) of pointers to its data blocks

- Logical view

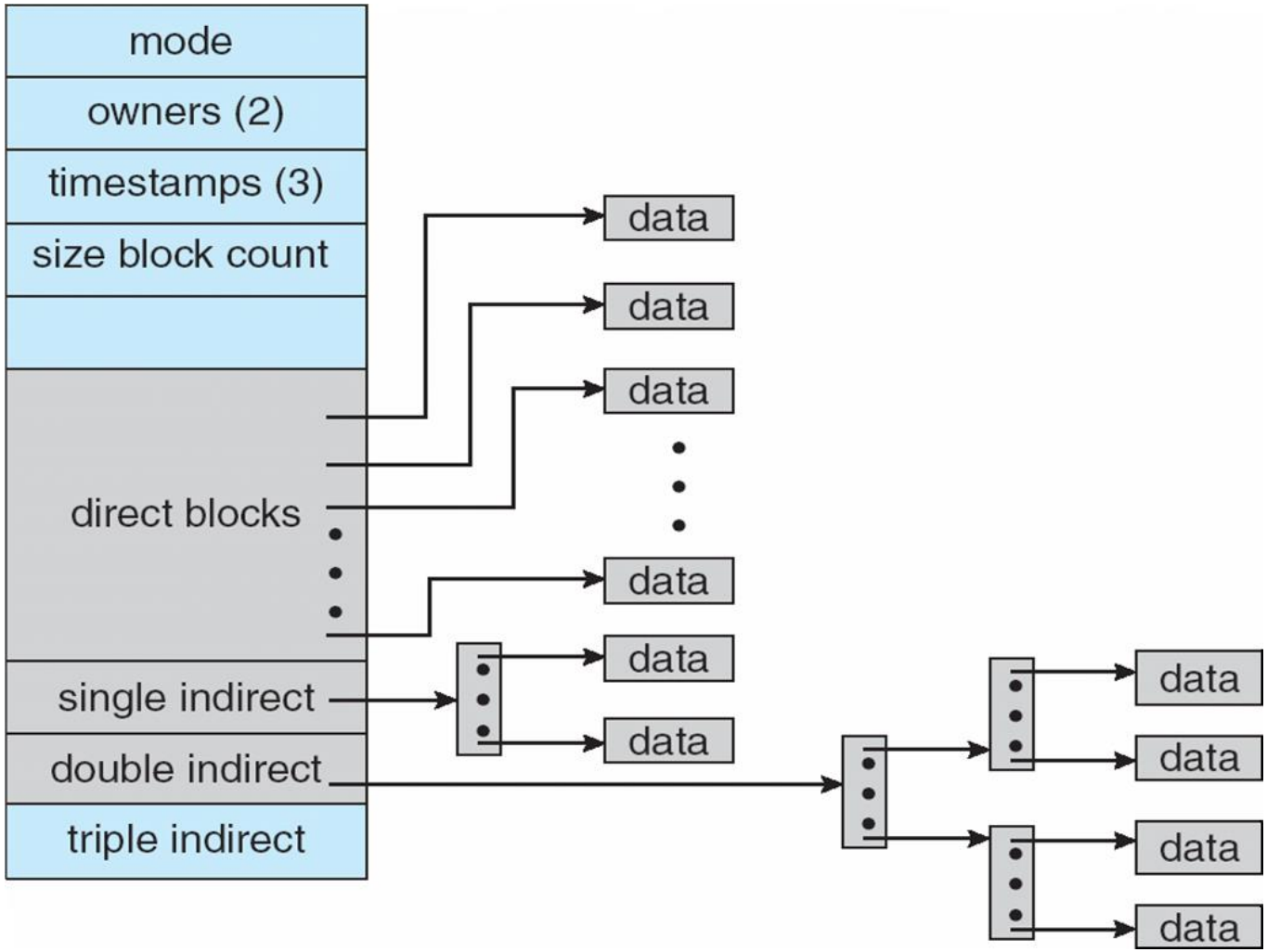


index table

Example of Indexed Allocation



Combined Scheme: UNIX UFS

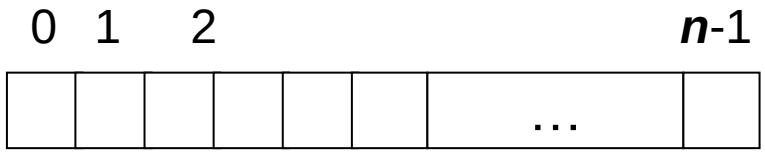


Performance

- Best method depends on file access type
  - Contiguous great for sequential and random
- Linked good for sequential, not random
- Declare access type at creation -> select either contiguous or linked
- Indexed more complex
  - Single block access could require 2 index block reads then data block read
  - Clustering can help improve throughput, reduce CPU overhead

Free-Space Management

- File system maintains **free-space list** to track available blocks/clusters
  - (Using term "block" for simplicity)
- **Bit vector** or **bit map** (***n*** blocks)



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

Free-Space Management (Cont.)

- Bit map requires extra space
  - Example:

    block size = 4KB = 2^{12} bytes

    disk size = 2^{40} bytes (1 terabyte)

    $n = 2^{40}/2^{12} = 2^{28}$ bits (or 32MB)

    if clusters of 4 blocks -> 8MB of memory

- Easy to get contiguous files

Indexed Free-Space

- Treat free space as a file
- Use index table
- Index based on variable size portions rather than block

Linked Free Space List on Disk

- Linked list (free list)
  - Cannot get contiguous space easily
  - No waste of space
  - No need to traverse the entire list (if # free blocks recorded)