

Operating Systems

Prepared by Dr. Alaa Mohamed

Module 6

جامعة الجميع الذكية

تسعى لتكون الكيان الأول الذي يتبنى مبدأ المجتمع التعليمي
التعاوني ويقدم العلم مقابل العمل لتنمية وبناء المجتمعات.

File Systems



جامعة الجميع الذكية
EVERYONE'S SMART UNIVERSITY

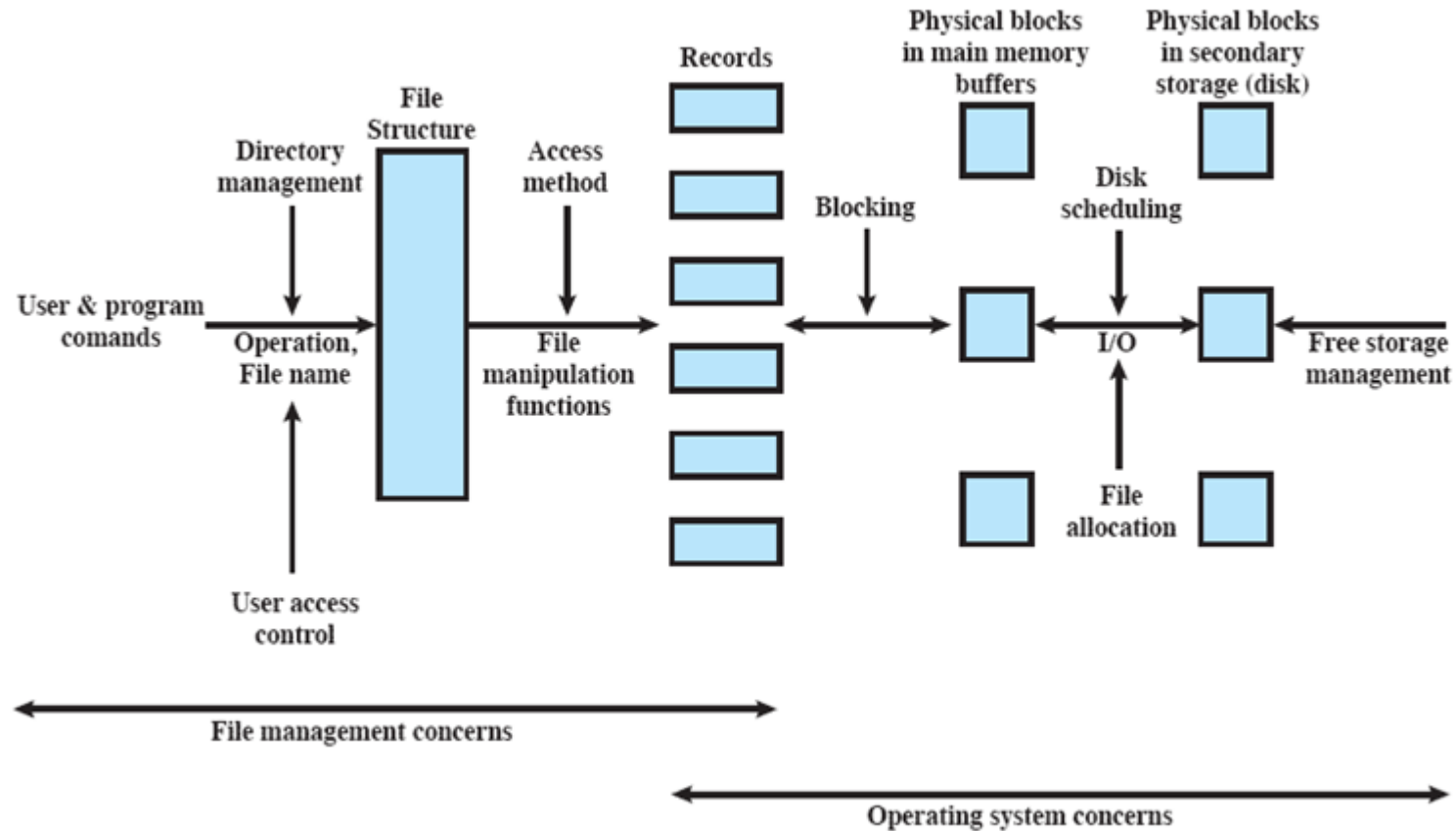
- Objectives:
 - File organisation and Access
 - File Directories
 - File Sharing
 - Record Blocking
 - Secondary Storage Management

- Files are the central element to most applications
- The File System is one of the most important part of the OS to a user
- Desirable properties of files:
 - Long-term existence
 - Sharable between processes
 - Structure
- File management system consists of system utility programs that run as privileged applications
- Concerned with secondary storage

- File systems also provide functions which can be performed on files, typically:
 - **Create:** A new file is defined and positioned within the structure of files.
 - **Delete:** A file is removed from the file structure and destroyed.
 - **Open:** An existing file is declared to be “opened” by a process, allowing the process to perform functions on the file.
 - **Close:** The file is closed with respect to a process, so that the process no longer may perform functions on the file, until the process opens the file again.
 - **Read:** A process reads all or a portion of the data in a file.
 - **Write:** A process updates a file, either by adding new data that expands the size of the file or by changing the values of existing data items in the file.

- A file management system is the set of system software that provides services to users and applications in the use of files.
 - Typically, the only way that a user or application may access files is through the file management system.
- Programmer does not need to develop file management software

- Meet the data management needs of the user
- Guarantee that the data in the file are valid
- Optimize performance
- Provide I/O support for a variety of storage device types
- Minimize lost or destroyed data
- Provide a standardized set of I/O interface routines to user processes
- Provide I/O support for multiple users (if needed)



Before performing any operation, the file system must identify and locate the selected file.

- This requires the use of some sort of directory that serves to describe the location of all files, plus their attributes.
- In addition, most shared systems enforce user access control

The basic operations that a user or application may perform on a file are performed at the record level.

- The user or application views the file as having some structure that organizes the records, such as a sequential structure

Elements of File management systems

The secondary storage must be managed.

- This involves allocating files to free blocks on secondary storage and managing free storage so as to know what blocks are available for new files and growth in existing files.
- In addition, individual block I/O requests must be scheduled

Disk scheduling and file allocation are both concerned with optimizing performance.

The optimization will depend on the structure of the files and the access patterns.

- File Management Referring to the logical structure of records
 - Physical organization discussed later
- Determined by the ***way*** in which files are accessed

- Important criteria include:
 - Short access time
 - Ease of update
 - Economy of storage
 - Simple maintenance
 - Reliability
- Priority will differ depending on the use (e.g. read-only CD vs Hard Drive)
 - Some may even conflict

- Many exist, but usually variations of:
 - Pile
 - Sequential file
 - Indexed sequential file
 - Indexed file
 - Direct, or hashed, file

- Contains information about files
 - Attributes
 - Location
 - Ownership
- Directory itself is a file owned by the operating system
- Provides mapping between file names and the files themselves

- File Name
 - Name as chosen by creator (user or program).
 - Must be unique within a specific directory.
- File type
- File Organisation
 - For systems that support different organizations

- Volume
 - Indicates device on which file is stored
- Starting Address
- Size Used
 - Current size of the file in bytes, words, or blocks
- Size Allocated
 - The maximum size of the file

- Date Created
- Identity of Creator
- Date Last Read Access
- Identity of Last Reader
- Date Last Modified
- Identity of Last Modifier
- Date of Last Backup
- Current Usage
 - Current activity, locks, etc

- The method for storing the previous information varies widely between systems
- Simplest is a list of entries, one for each file
 - Sequential file with the name of the file serving as the key
 - Provides no help in organizing the files
 - Forces user to be careful not to use the same name for two different files

- A directory system should support a number of operations including:
- **Search:** When a user or application references a file, the directory must be searched to find the entry corresponding to that file.
- **Create file:** When a new file is created, an entry must be added to the directory.

- **Delete file:** When a file is deleted, an entry must be removed from the directory.
- **List directory:** All or a portion of the directory may be requested. Generally, this request is made by a user and results in a listing of all files owned by that user, plus some of the attributes of each file
- **Update directory:** Because some file attributes are stored in the directory, a change in one of these attributes requires a change in the corresponding directory entry.

- One directory for each user and a master directory
 - Master directory contains entry for each user
 - Provides address and access control information
- Each user directory is a simple list of files for that user
 - Does not provide structure for collections of files

- Master directory with user directories underneath it
- Each user directory may have subdirectories and files as entries

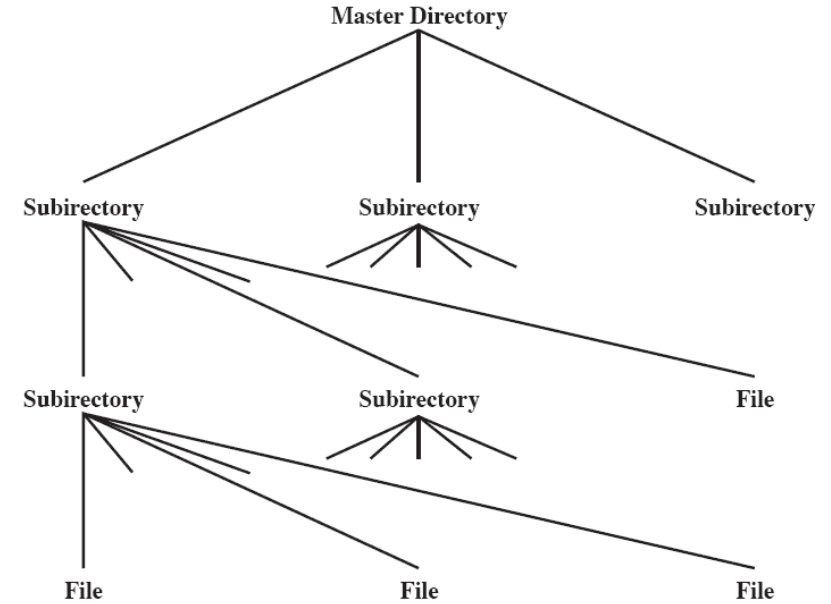
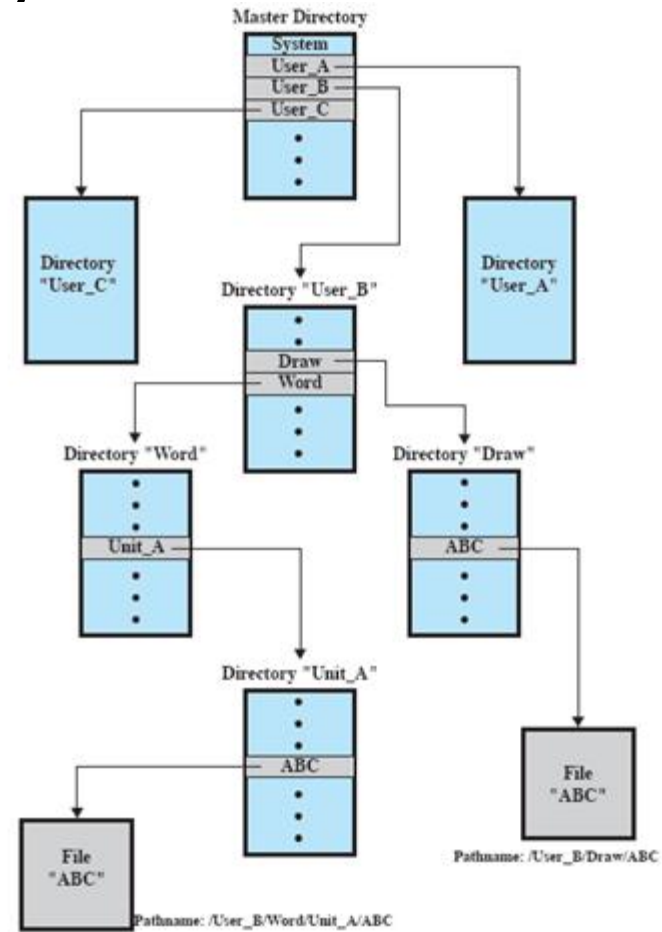


Figure 12.4 Tree-Structured Directory

- Users need to be able to refer to a file by name
 - Files need to be named uniquely, but users may not be aware of all filenames on a system
- The tree structure allows users to find a file by following the directory path
 - Duplicate filenames are possible if they have different pathnames

Example of Tree-Structured Directory



- In multiuser system, allow files to be shared among users
- Two issues
 - Access rights
 - Management of simultaneous access

- A wide variety of access rights have been used by various systems
 - often as a hierarchy where one right implies previous
- None
 - User may not even know of the files existence
- Knowledge
 - User can only determine that the file exists and who its owner is

- Execution
 - The user can load and execute a program but cannot copy it
- Reading
 - The user can read the file for any purpose, including copying and execution
- Appending
 - The user can add data to the file but cannot modify or delete any of the file's contents

- Updating
 - The user can modify, delete, and add to the file's data.
- Changing protection
 - User can change access rights granted to other users
- Deletion
 - User can delete the file

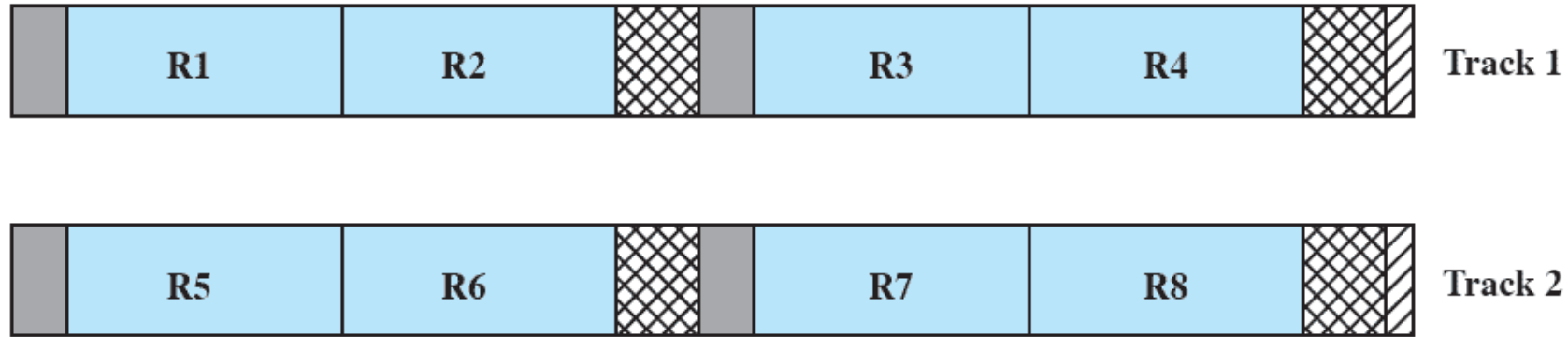
- Owner
 - Usually the files creator, usually has full rights
- Specific Users
 - Rights may be explicitly granted to specific users
- User Groups
 - A set of users identified as a group
- All
 - everyone

- User may lock entire file when it is to be updated
- User may lock the individual records during the update
- Mutual exclusion and deadlock are issues for shared access

- Records are the logical unit of access of a structured file
 - But blocks are the unit for I/O with secondary storage
- Three approaches are common
 - Fixed length blocking
 - Variable length spanned blocking
 - Variable-length unspanned blocking

- Fixed-length records are used, and an integral number of records are stored in a block.
- Unused space at the end of a block is *internal fragmentation*

Fixed Blockings



Fixed Blocking



Data



Gaps due to hardware design



Waste due to block fit to track size



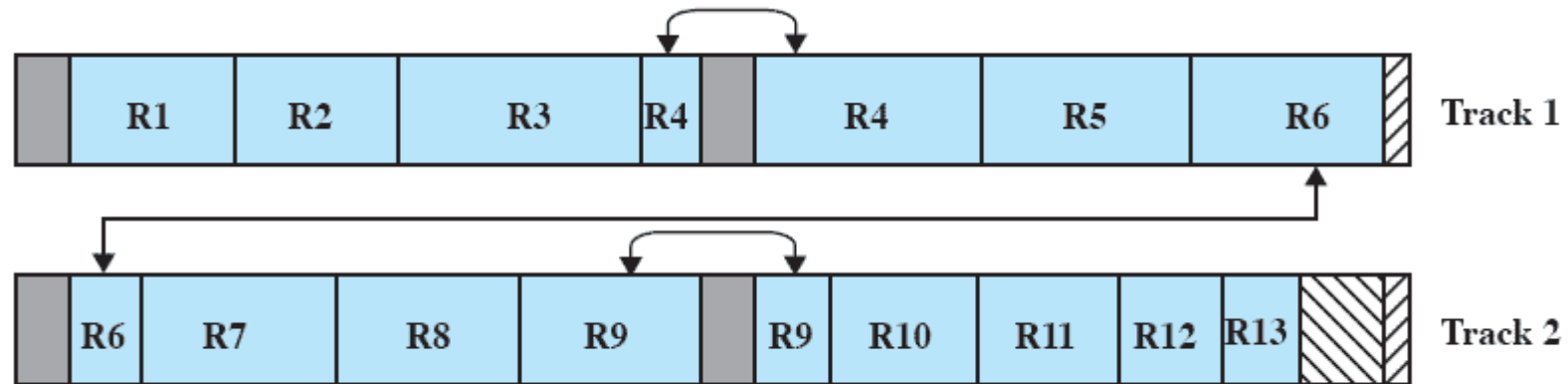
Waste due to record fit to block size



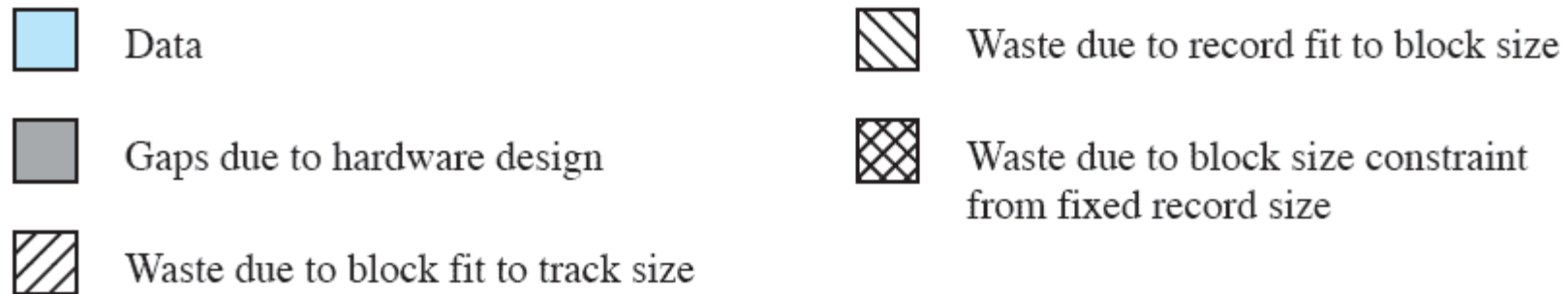
Waste due to block size constraint
from fixed record size

- Variable-length records are used and are packed into blocks with no unused space.
- Some records may span multiple blocks
 - Continuation is indicated by a pointer to the successor block

Variable Blocking: Spanned

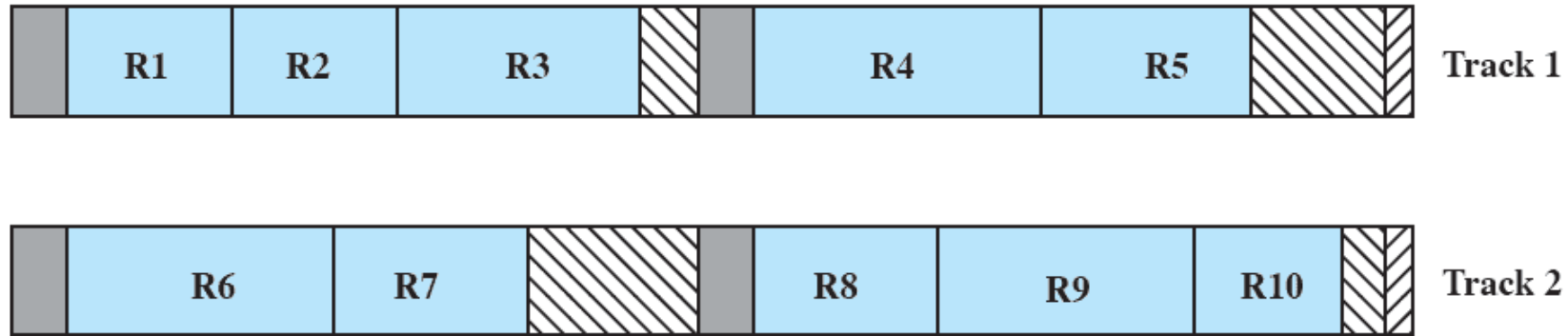


Variable Blocking: Spanned

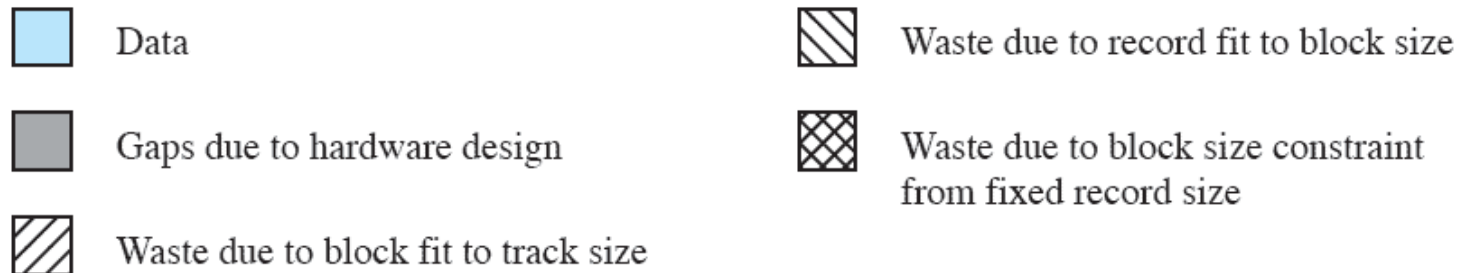


- Uses variable length records without spanning
- Wasted space in most blocks because of the inability to use the remainder of a block if the next record is larger than the remaining unused space.

Variable Blocking: Unspanned



Variable Blocking: Unspanned



- The Operating System is responsible for allocating blocks to files
- Two related issues
 - Space must be allocated to files
 - Must keep track of the space available for allocation

1. When a file is created – is the maximum space allocated at once?
2. Space is added to a file in contiguous 'portions'
 - What size should be the 'portion'?
3. What data structure should be used to keep track of the file portions?

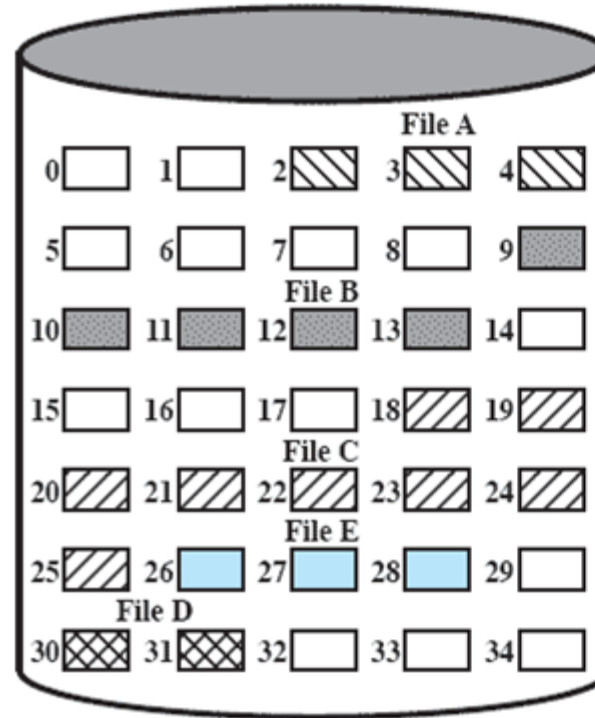
- Need the maximum size for the file at the time of creation
- Difficult to reliably estimate the maximum potential size of the file
- Tend to overestimated file size so as not to run out of space

- Two extremes:
 - Portion large enough to hold entire file is allocated
 - Allocate space one block at a time
- Trade-off between efficiency from the point of view of a single file, or the overall system efficiency

- Three methods are in common use:
 - contiguous,
 - chained, and
 - indexed.

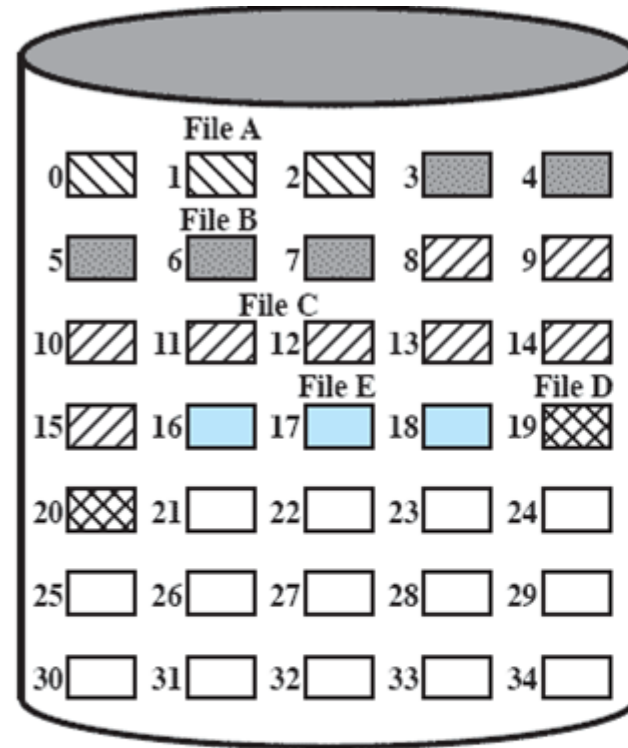


- Single set of blocks is allocated to a file at the time of creation
- Only a single entry in the file allocation table
 - Starting block and length of the file
- External fragmentation will occur
 - Need to perform compaction



File Allocation Table

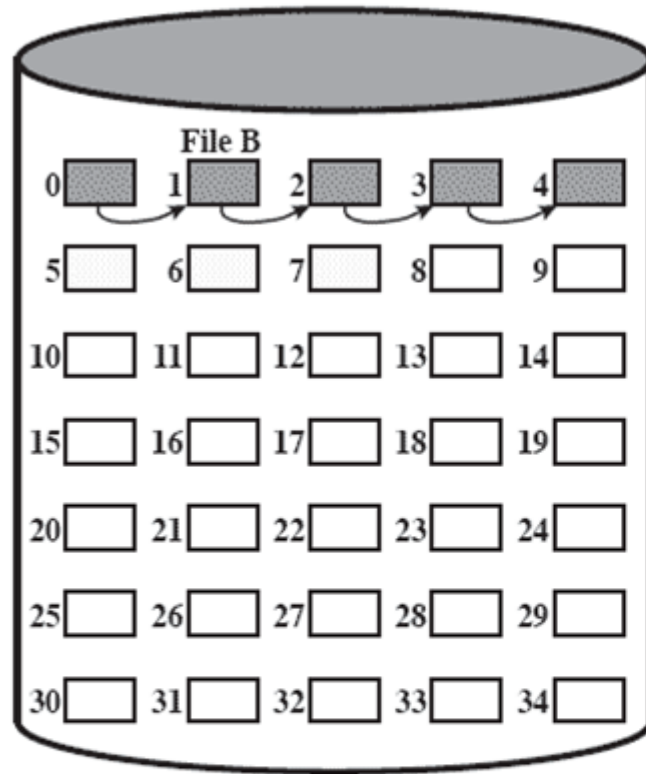
File Name	Start Block	Length
File A	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



File Allocation Table

File Name	Start Block	Length
File A	0	3
File B	3	5
File C	8	8
File D	19	2
File E	16	3

- Allocation on basis of individual block
- Each block contains a pointer to the next block in the chain
- Only single entry in the file allocation table
 - Starting block and length of file
- No external fragmentation
- Best for sequential files

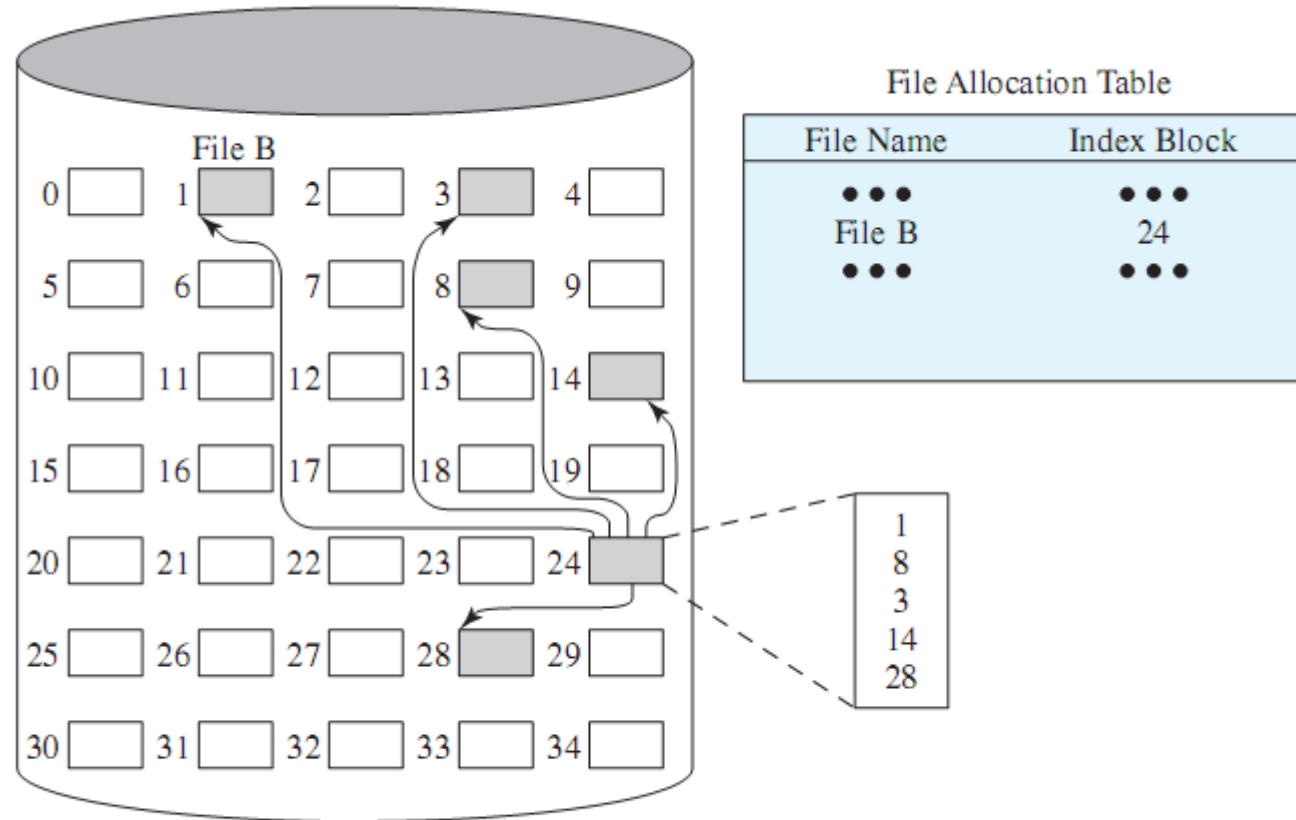


File Allocation Table

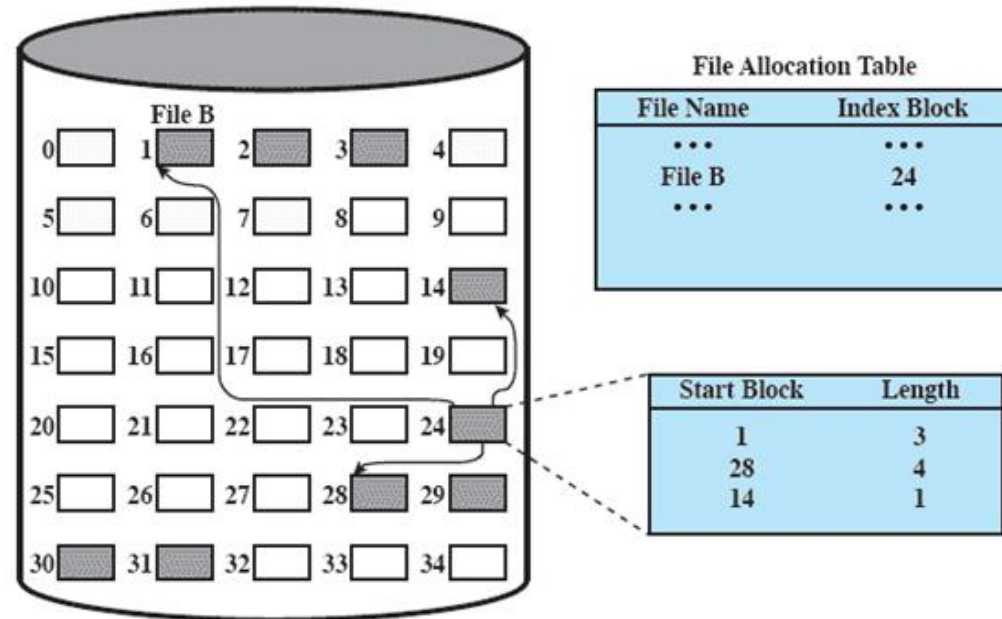
File Name	Start Block	Length
...	...	...
File B	0	5
...	...	...

- File allocation table contains a separate one-level index for each file
- The index has one entry for each portion allocated to the file
- The file allocation table contains block number for the index

- Allocation may be either
 - Fixed size blocks or
 - Variable sized blocks
- Allocating by blocks eliminates external fragmentation
- Variable sized blocks improves locality
- Both cases require occasional consolidation



Indexed Allocation with Block Portions



Indexed Allocation with Variable-Length Portions

End of Module 6

