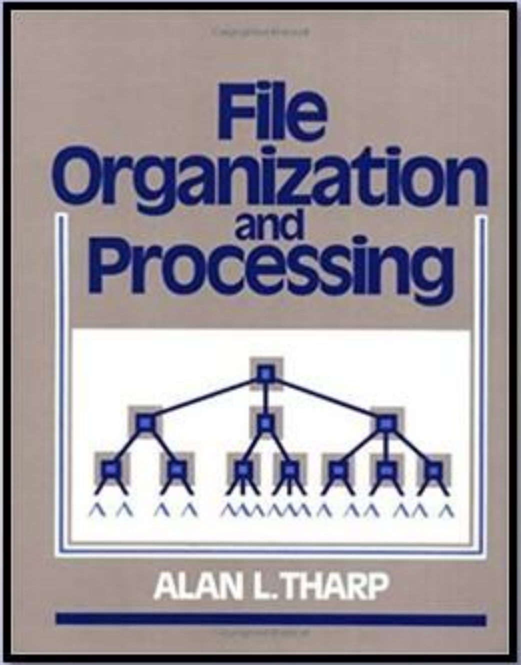
# File Organization & Processing

## CS2202

***Instructor:***

**Dr. Mohamed Hassan**

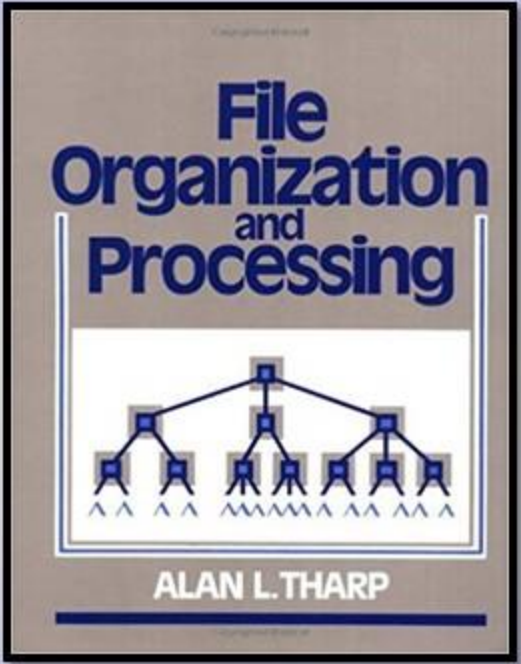


File Organization and Processing

ALAN L. THARP

# Chapter 3:

- File management & organization



# A journey of a Byte
# and
# Buffer Management

# A journey of a byte

- Suppose in our program we wrote:

```
outfile << c;
```

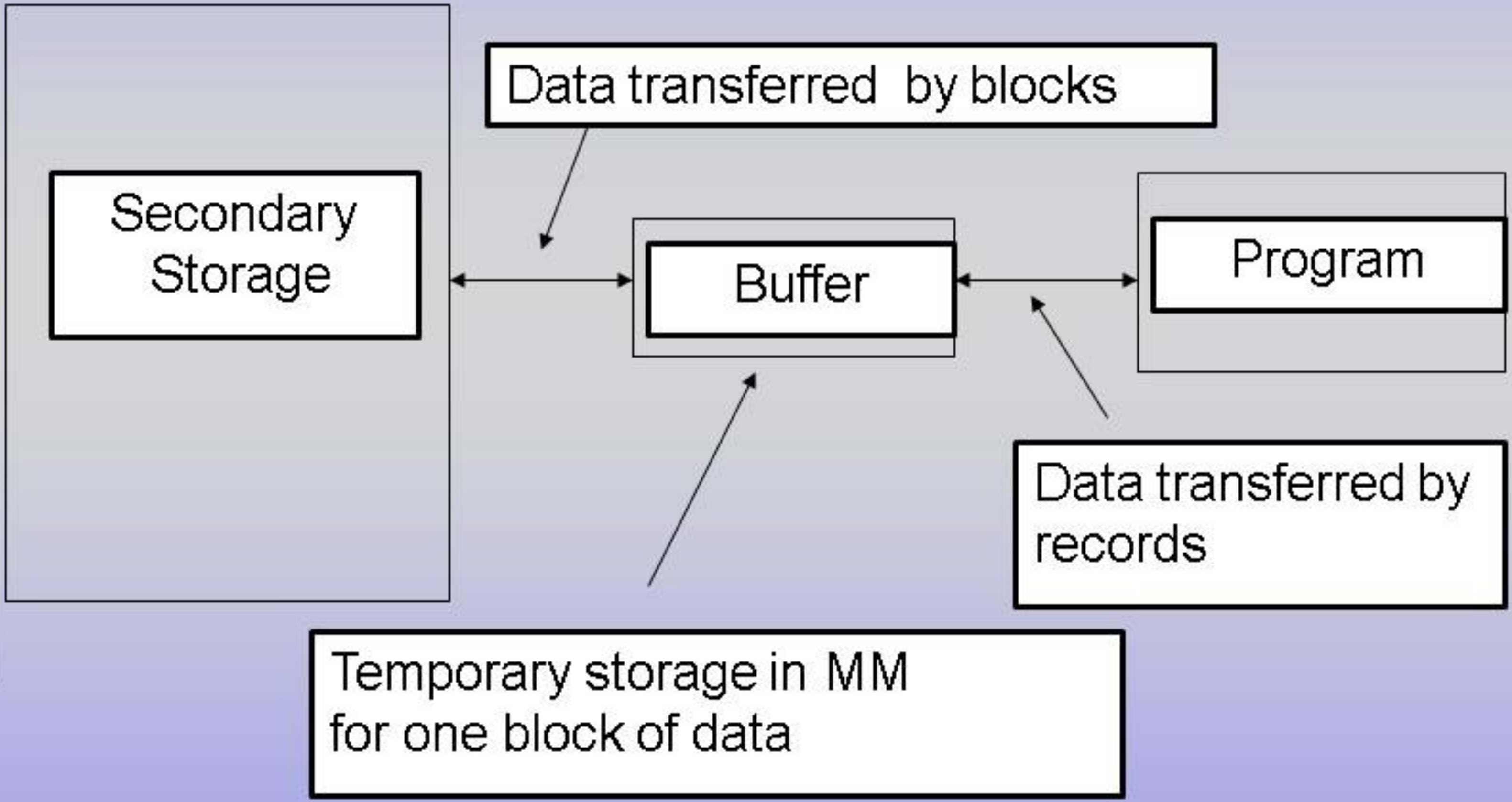
- This causes a call to the **file manager** (a part of O.S. responsible for I/O operations)
- The **O/S (File manager)** makes **sure that the byte is written to the disk.**
- **Pieces of software/hardware involved in I/O**:
  - Application Program
  - Operating System/ file manager
  - I/O Processor
  - Disk Controller

- **Application program**
  - Requests the I/O operation
- **Operating system / file manager**
  - Keeps tables for all opened files
  - Brings appropriate sector to buffer.
  - Writes byte to buffer
  - Gives instruction to I/O processor to write data from this buffer into correct place in disk.
  - Note: the buffer is an exact image of a cluster in disk.
- **I/O Processor**
  - a separate chip; runs independently of CPU
  - **Find a time when drive is available to receive data and put data in proper format for the disk**
  - Sends data to disk controller
- **Disk controller**
  - A separate chip; **instructs the drive to move R/W head**
  - **Sends the byte to the surface when the proper sector comes under R/W head.**

# Buffer Management

- **Buffering means working with large chunks of data in main memory so the number of accesses to secondary storage is reduced**.
- Today, we'll discuss the System I/O buffers. **These are beyond the control of application programs and are manipulated by the O.S**.
- Note that the **application program may implement its own "buffer**" – i.e. a place in memory (variable, object) that accumulates large chunks of data to be later written to disk as a chunk..

# System I/O Buffer



Secondary Storage

Buffer

Program

Data transferred by blocks

Data transferred by records

Temporary storage in MM
for one block of data

# Buffer Bottlenecks

- Consider the following program segment:

```
while (1) {
  infile >> ch;
   if (infile.fail()) break;
   outfile << ch;
}
```

- **What happens if the O.S. used only one I/O buffer?**

  ⇒ **Buffer bottleneck**

- **Most O.S. have an input buffer and an output buffer.**

# Buffering Strategies

- **Double Buffering:**

**Two buffers can be used to allow processing and I/O to overlap.**

- Suppose that a **program is only writing to a disk**.
- CPU **wants to fill a buffer at the same time that I/O is being performed.**
- If two buffers are used and I/O-CPU overlapping is permitted, CPU can be filling one buffer while the other buffer is being transmitted to disk.
- When both tasks are finished, the roles of the buffers can be exchanged.

- The actual management is done by the O.S.

# Other Buffering Strategies

- **Multiple Buffering**: **instead of two buffers any number of buffers can be used to allow processing and I/O to overlap**.
- **Buffer pooling:**
  - There is a pool of buffers.
  - When a request for a sector is received, O.S. first looks to see that sector is in some buffer.
  - If not there, it brings the sector to some free buffer. If no free buffer exists, it must choose an occupied buffer. (usually LRU strategy is used)

# File Management and organization

- **A file** is a **named entity used to save results from a program or provide data to a program**. Access control is enforced generally on file level.
- **File Management System** is a **set of system software that provides services related to use of files** (e.g. copying, creating, deleting, naming etc.)

# Terms Used with Files

- **Field**
  - Basic element of data
  - Contains a single value
  - Characterized by its length and data type
- **Record**
  - Collection of related fields
  - Treated as a unit
    - Example: employee record

# Terms Used with Files

- **File**
  - Collection of similar records
  - Treated as a single entity
  - Have unique file names
  - May restrict access
- **Database**
  - Collection of related data
  - Relationships exist among elements

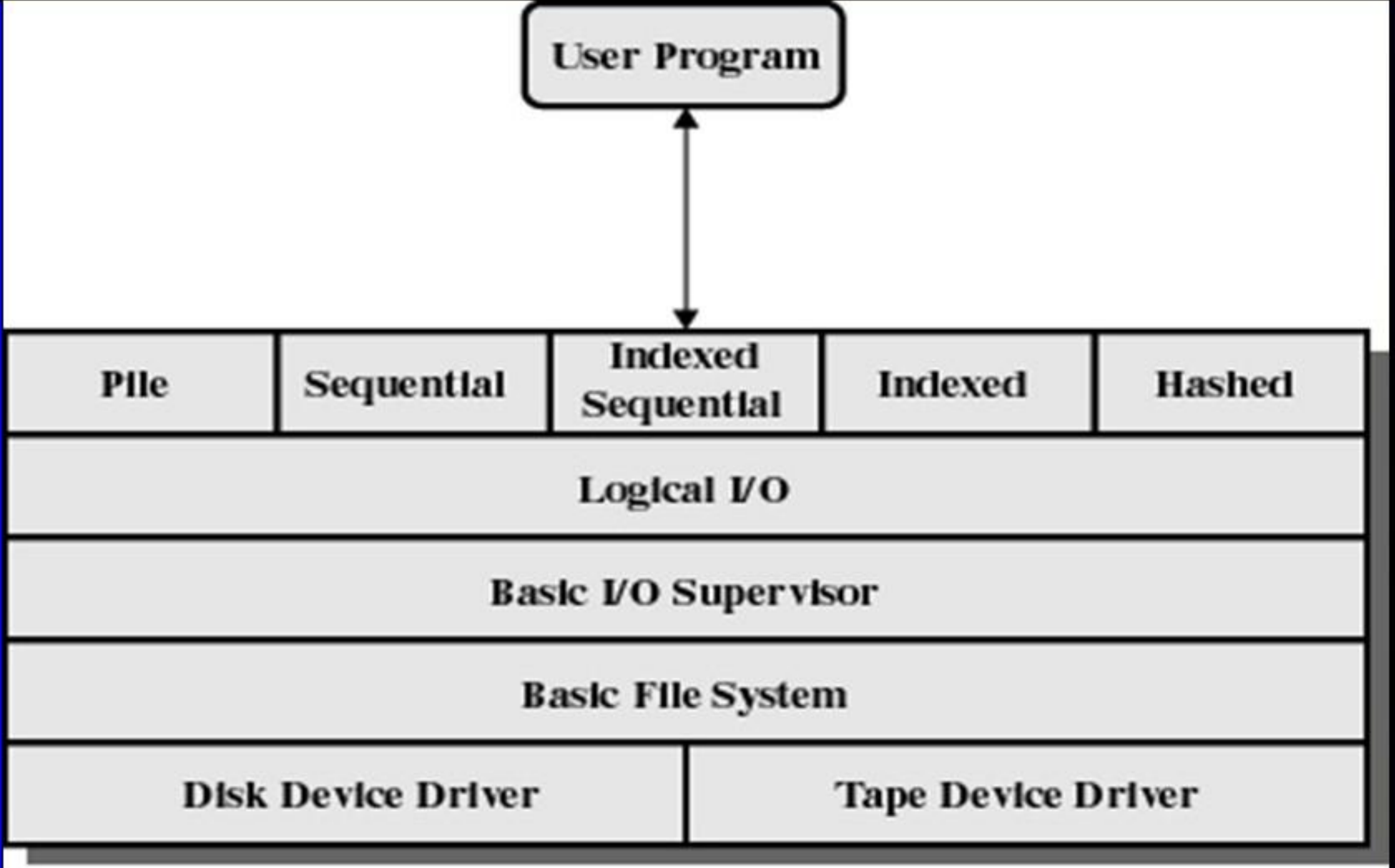


Figure 12.1 File System Software Architecture [GROS86]

# Device Drivers

- Lowest level
- Communicates directly with peripheral devices
- **Responsible for starting I/O operations on a device**
- **Processes the completion of an I/O request**

# Basic File System

- Physical I/O
- Deals with exchanging blocks of data
- **Concerned with the placement of blocks**
- **Concerned with buffering blocks in main memory**

# Basic I/O Supervisor

- **Responsible for file I/O initiation and termination**
- Control structures are maintained
- **Concerned with scheduling access to optimize performance**
- **Part of the operating system**

# Logical I/O

- **Enables users and applications to access records**
- Provides general-purpose record I/O capability
- **Maintains basic data about file**

# Types of file organization

- **There are four types of file organization that you need to know about:**
- **Serial**
- **Sequential**
- **Indexed Sequential**
- **Direct /Random Access**

# The Pile

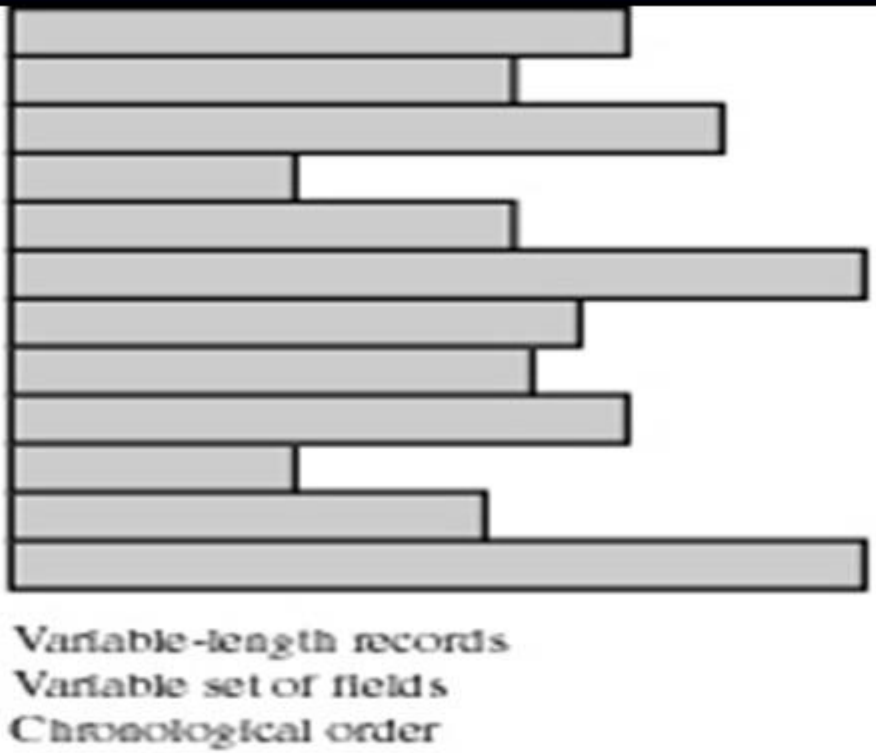
- is one **in which the records have been stored in the order in which they have arisen. They have not been sorted into any particular order.**
  - Data are collected in the order they arrive
  - Purpose is to accumulate a mass of data and save it
  - Records may have different fields
  - No structure
  - Record access is by exhaustive search
- A shopping list is an example of a non-computerised serial file.
  - A collection of records
  - No particular sequence

# The Pile

- An example of a serial file is an unsorted transaction file (more on this in a minute ☺).

- Cannot be used as master
- Used as temporary transaction file
- Records stored in the order received

# Pile



(a) Pile File

**Figure 12.3 Common File Organizations**

# Sequential File Organization

- A sequential file is one in which the records are stored in sorted order of one or more key fields.

# File Organization

- The Sequential File
  - **Fixed format** used for records
  - Records are the **same length**
  - All fields the same (order and length)
  - Field names and lengths are attributes of the file
  - **One field is the key filed**
    - **Uniquely** identifies the record
    - Records are stored in **key sequence**
- **Adding/deleting record requires making new file** (so that the sequence is maintained)
- **Used as master files**

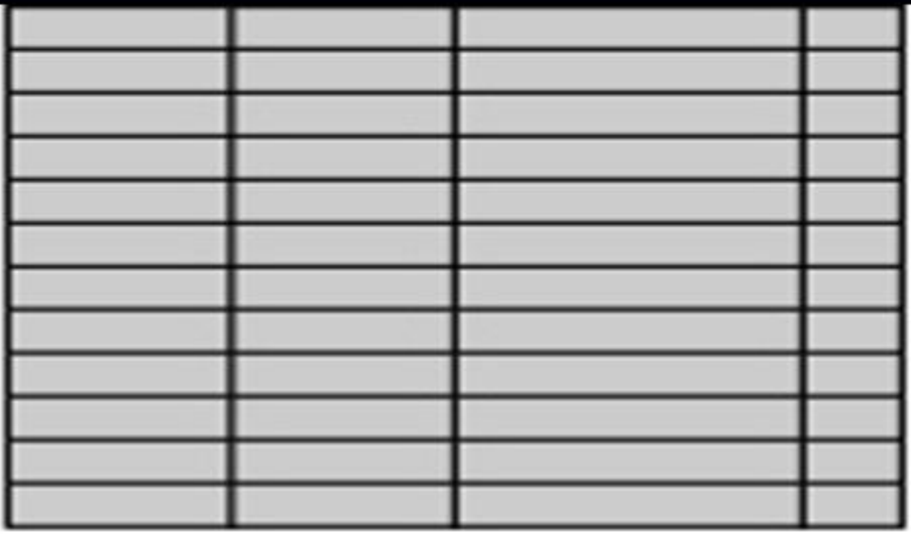
# Sequential File Organization

- Sequential access means that data is accessed in a predetermined, ordered sequence.
- Sequential access is sometimes the only way of accessing the data, for example if it is on a tape.

# File Organization

- The Sequential File
  - **New records** are **placed in a log file or transaction file**
  - **Batch update** is performed **to merge the log file with the master file**

# Sequential File

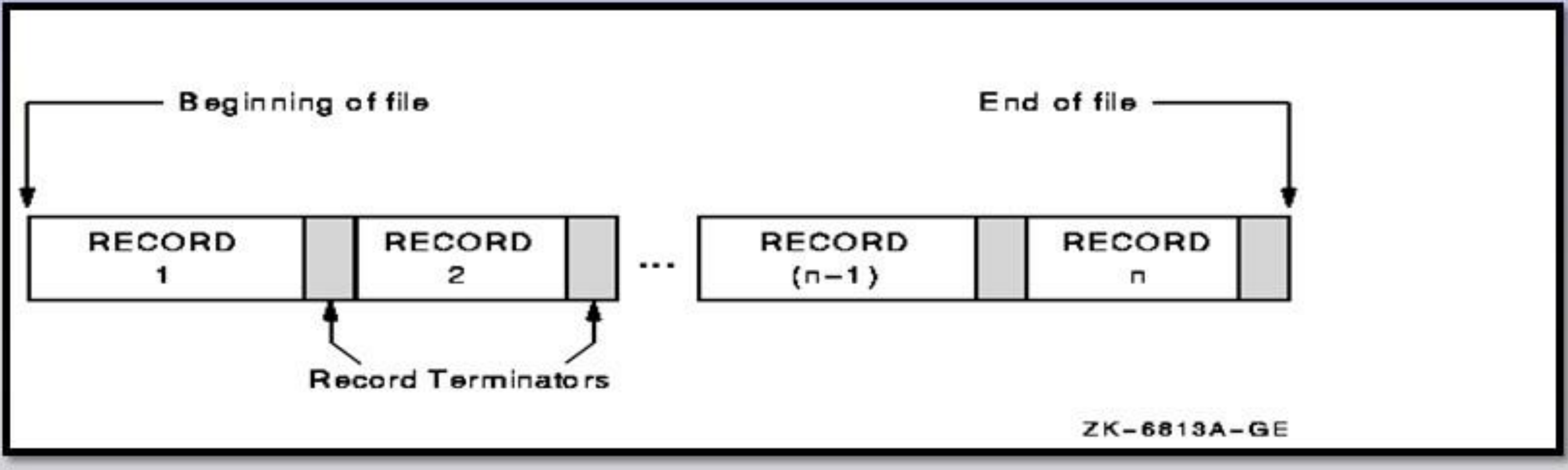


Fixed-length records
Fixed set of fields in fixed order
Sequential order based on key field

(b) Sequential File

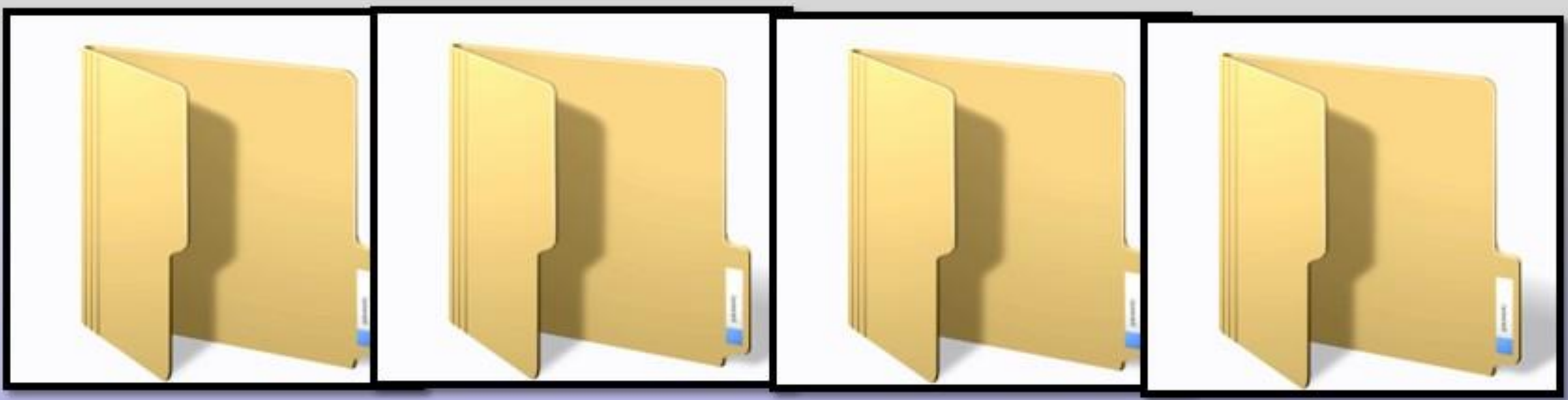
Figure 12.3 Common File Organizations

# Sequential file

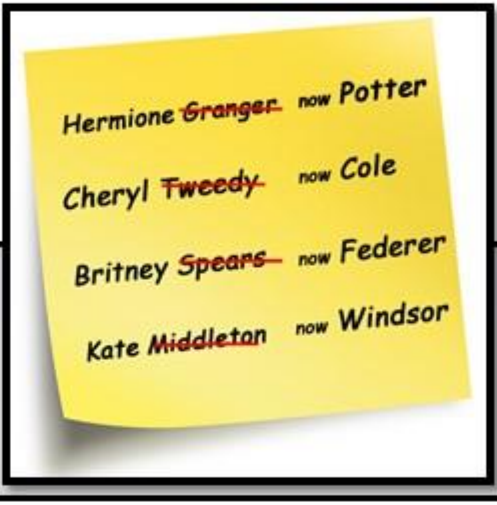


# Master files & transaction files

- Serial files are often used as transaction files.
- Sequential files are used as master files.
  - – A company's master file might hold all the data about every employee



A **transaction** file might hold a list of all the employees who have gotten married this month and changed their names.



Hermione ~~Granger~~ now Potter

Cheryl ~~Tweedy~~ now Cole

Britney ~~Spears~~ now Federer

Kate ~~Middleton~~ now Windsor

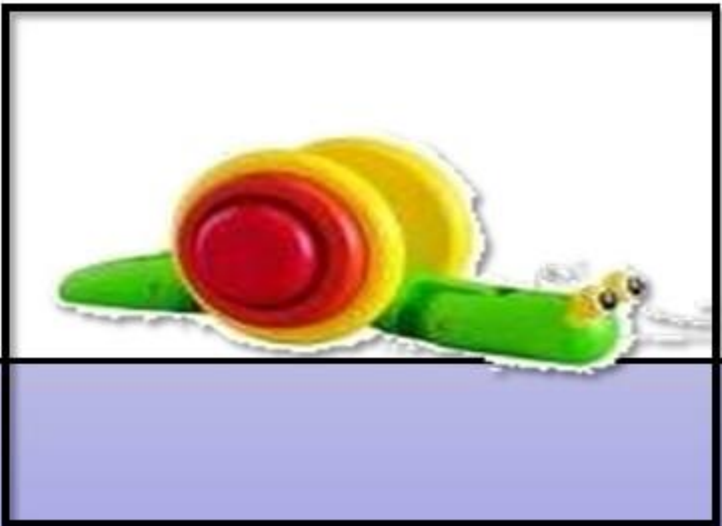
- **The master file would be read one record at a time**
- **The transaction file would be used to update the master file**

# Advantages

- **Simple** file design
- **Very efficient** when **most of the records must be processed** e.g. Payroll
- **Very efficient** if **the data has a natural order**
- **Can be stored on inexpensive** devices like magnetic tape.

# Disadvantages

- **Entire file must be processed even if a single record is to be searched.**
- **Transactions have to be sorted before processing**
- **Overall processing is slow**, because **you have to go through each record until you get to the one you want**!



# Indexed sequential file

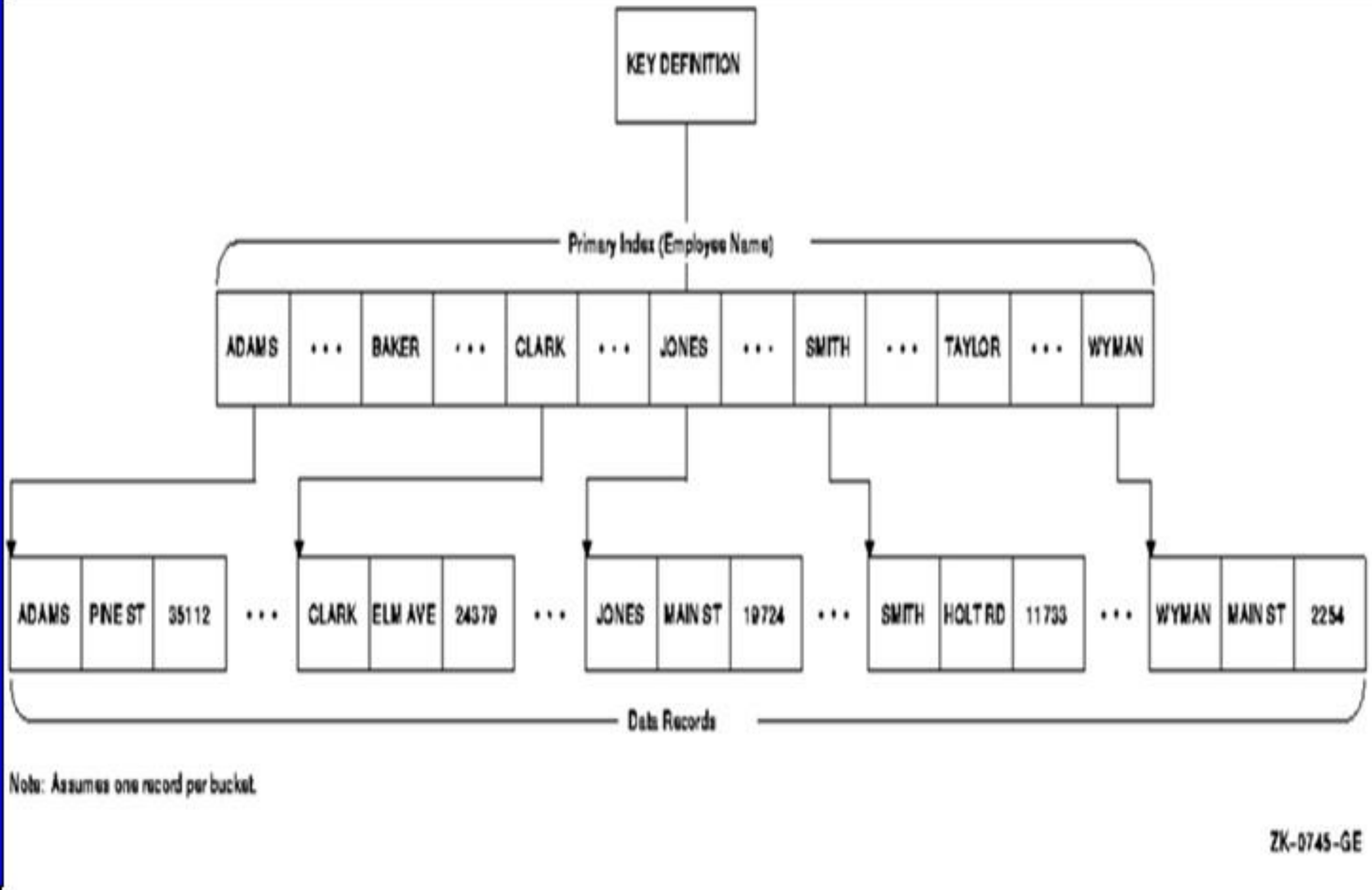
- **Each record of a file has a key field which uniquely identifies that record**.
- An **index** consists **of keys and addresses**, just like an index in a book:
  - The pages in a book are stored sequentially, so you can read through it page by page
  - OR You can look up the page you want
  - in the index and flick straight to it

**Because each record has an index, we can access individual records directly, without having to scroll through all the other records first**

# Indexed sequential file

- An indexed sequential file is a sequential file (i.e. sorted into order of a key field) which has an index.
- A full index to a file is one in which there is an entry for every record.
- **Because each record has an index, we can access individual records directly, without having to scroll through all the other records first**.

# Indexed sequential file



## Indexed sequential file

- Indexed sequential files are important for applications where data needs to be accessed.....
  - sequentially , one record after another
  - OR
  - randomly using the index.

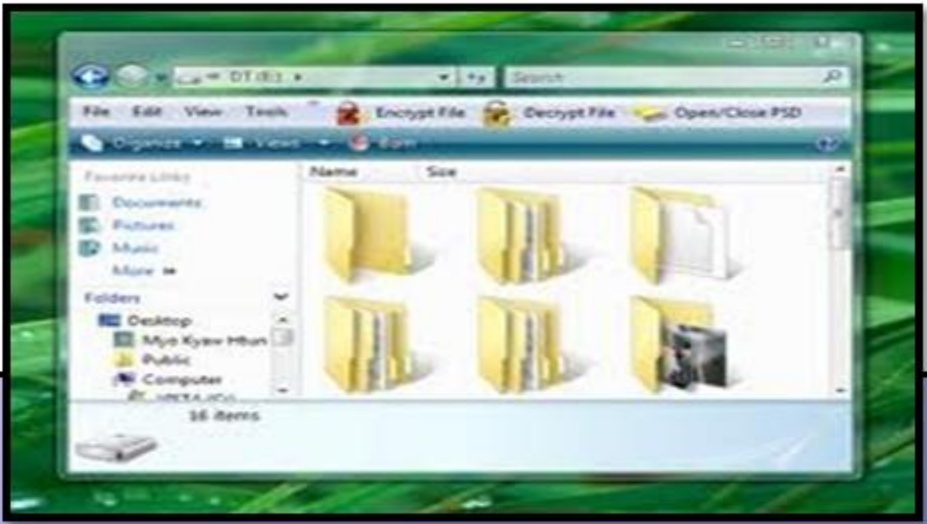
# An example of an Indexed Sequential file

- A company may store details about its employees as an indexed sequential file. Sometimes the file is accessed....
- **sequentially**. For example **when the whole of the file is processed to produce pay slips at the end of the month**.

- **randomly**. Maybe an **employee changes address, or a female employee gets married and changes her surname**

# Indexed sequential file

- An **indexed sequential file** **can only be stored on a random access device e.g. magnetic disc or CD.**
- This is **because we need a device that will allow us direct access to random files**, rather than **the sequential access that magnetic tape allows.**

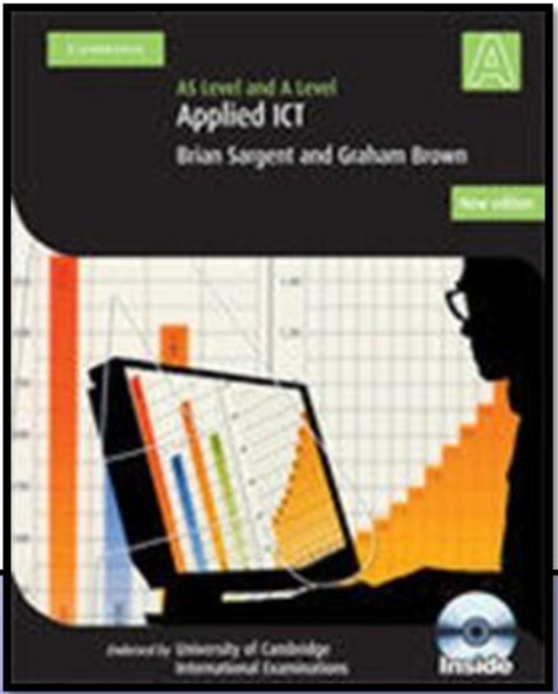


# Advantages

- Provides **flexibility** for **users who need both type of access with the same file**
- **Faster than sequential**

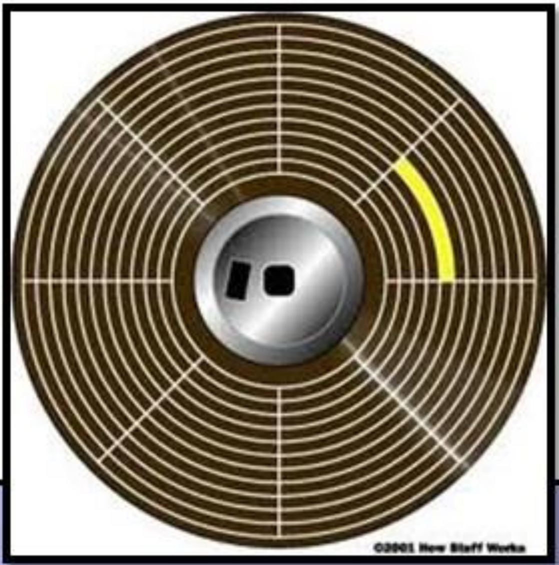
# Disadvantages

- **Extra storage space for the index is required**, just like in a book: your text book would be 372 pages without the index (go on, check!) but is 380 pages with the index.


Cambridge
A
AS Level and A Level
Applied ICT
Brian Sargent and Graham Brown
New edition
endorsed by University of Cambridge International Examinations
Inside

# Direct (Random) File Organization

- **Records are read directly from or written on to the file.**
- The **records** are **stored at known address**.
- The **address** is **calculated by applying a mathematical function to the key field.**


©2001 How Stuff Works

# Direct (Random) File Organization

- A **random file** would **have to be stored on a direct access backing storage medium e.g. magnetic disc, CD, DVD**
- Example : Any information retrieval system. Eg Train timetable system.

## Advantages

- Any **record** can be **directly accessed**.
- **Speed** of record **processing is very fast**.
- **Up-to-date** file because **of online updating**.
- **Concurrent** processing is **possible**.

# Disadvantages

- More **complex** than sequential
- Does not fully use memory locations
- More **security and backup problems**