5.1 System Software OS Notes

Objective:

- Explain why a computer system requires an Operating System (OS).
- Explain key management tasks carried out by Operating System Including memory management, file management, security management, hardware management (input/output/peripherals), process management.
- Show understanding of need for typical **Utility Software** provided with an Operating System Including disk formatter, virus checker, defragmentation software, disk contents analysis/disk repair software, file compression, back-up software.
- Show understanding of **Program Libraries** including: • Software under development is often constructed using existing code from program libraries • Benefits to developer of software constructed using library files, including Dynamic Link Library (DLL) files.

Why a computer system requires an Operating System (OS).

- It provides user interface and hides complexity of hardware from user. Hardware is unusable without an OS.
- It controls overall working of computer system.
- It control all attached hardware devices and efficiently use computer resources (processor, memory, input / output devices).
- Provides platform or environment on which other programs can be run.
- It control how hardware devices works and communicate with each other.

System Software

System Software is designed to **control** and **work** with computer **hardware**.

Types of System Software:

- **Operating System** is set of programs designed to run in background on computer system, giving an environment in which **Application Software** can be executed.
  Example: Mac OS , Linux, Windows, Android, iOS
- A **Utility** or **Software Utility** is computer system software intended to analyze, configure, monitor, or help to maintain a computer.

Operating system and utility software typically depend on each other to function properly.

Operating System Tasks

- **User System Interface:**

User interface is needed to allow user to get software and hardware to do something useful.

Types of User System Interface

❖ **Command-line Interface (CLI):**

OS provides a method of interaction that is **non-graphical** called command line interface (CLI). This is **text-only service** with feedback from OS appearing in text.
Using CLI, requires **knowledge** of commands available on a particular machine.

- **Graphical user interface (GUI).**

OS also provides an environment with tiles, icons and menus. This type of interface is called the graphical user interface (GUI) because user interacts with images through a mouse, keyboard or touchscreen.

Program-Hardware Interface:

Program development tools allow programmer to write program without needing to know details of how hardware, particularly processor, actually works. Operating system provide mechanism for execution of **developed program** and ensure that hardware does what software wants it to do.

Resource Management:

Resource management provided by operating system aims to achieve **optimum efficiency** in computer system use. Two important aspects of this are:

- **Scheduling** of processes
- Resolution of **conflicts** when two processes require the same resource.

Memory Management:

Memory management is management of a computer's **main memory**.

Three aspects of Memory Management:

- **Memory Protection**

Memory protection ensures that two **competing applications** cannot use **same memory** locations at same time. If this was not done, data could be lost, applications could produce **incorrect results**, there could be **security issues**, or computer may crash.

- **Memory Organization** scheme is used to **achieve optimum** usage of **limited** memory size. This can be done with use of

✓ **Single (contiguous) allocation**, where all of the memory is made available to a single application.

✓ **Partitioned allocation**, where memory is split up into **contiguous partitions** and memory management then allocates a partition to an application.

✓ **Paged memory**, which is similar to partitioned allocation, but each partition is of a **fixed size**. This is used by virtual memory systems

- **Memory Usage Optimization**

Memory optimization is used to determine how computer memory is **allocated** and **deallocated** when a number of applications are running simultaneously. This involves decisions about which processes should be in **main memory** at any one time and where they are stored in this memory.

Security Management:

Function of security management is to ensure **integrity, confidentiality** and **availability of data**. This can be achieved by

- Carrying out **operating system updates** as and when they become available.
- Ensuring that **antivirus software** is always **up-to-date**.
- Communicating with a **firewall** to check **all traffic** to and from computer.
- Making use of privileges to prevent users entering **'private areas'** on a computer which permits **multi-user activity**. This helps to ensure **Privacy of Data**.
- Maintaining **Access Rights** for all users.
- Offering ability for **Recovery** of data (system restore) when it has been lost or corrupted
- Helping to prevent **illegal intrusion** to computer system.

Process Management:

Process is a **program** which is being run on a computer. Process management involves **allocation** of **Resources** and permits **sharing** and **exchange** of data, thus allowing all processes to be fully **synchronized** (by scheduling of resources, resolution of software conflicts and so on).

Hardware Management:

Hardware management involves all **Input** and **Output** peripheral devices. **Functions of hardware management** include

- Communicating with all input and output devices using **device drivers.**
- **Translating data** from a file (defined by OS) into a format that input/output device can understand using **device drivers.**
- Ensuring each hardware resource has **priority** so that it can be **used** and **released** as required.

Hardware Devices Management is essentially control and management of **queues** and **buffers.**

For example, when printing out a document, printer management

- Locates and loads the **printer driver** into memory.
- Sends data to a **printer buffer** ready for printing.
- Sends data to **printer queue** (if printer is busy) before sending to printer buffer.
- Sends various **control commands** to printer throughout printing process.
- Receives and handles **error messages** and **interrupts** from printer.

File Management:

Main tasks of file management include

- Defining **file naming** conventions which can be used (filename.docx, where extension can be .pdf, .txt, .xls, and so on)
- Performing **specific tasks**, such as create, open, close, delete, rename, copy, move
- Maintaining **Directory Structures**.
- Ensuring **Access Control** mechanisms are maintained, such as **Access rights** to files, password protection, making files available for editing, and so on.
- Specifying **logical file storage** format (such as FAT or NTFS), depending on which type of **disk formatter** is used.

• Ensuring **Memory Allocation** for file by reading it from HDD/SSD and loading it into memory.

Utility Software

- **It is system software designed to analyze, configure, optimize or maintain computer.**
- **Utility Program** might be provided by OS but it might be installed as **Separate entity**.
- Utility software is usually initiated by user, but some, such as virus checkers, can be set up to constantly run in background.

Example of Utility Program

Hard Disk Formatter And Checker:

Hard disk formatter carry out following tasks:

- Removing **Existing Data** from a disk.
- Setting up **File System** on Hard disk, based on a **table of contents** that allows a file recognized by OS to be associated with a specific physical part of disk.
- Partitioning disk into **Logical Drives** if required.

Hard Drive Checker:

Utility program, which might be a component of a disk formatter, performs **Disk Contents Analysis** and **Disk Repair** when needed. Program first checks for errors on disk. Some errors arise, resulting in defect what is called a '**Bad Sector**'.

Causes of Bad Sectors:

- Improper **shutdown** of Windows.
- Defects of hard disk, including general surface wear, pollution of air inside unit, or head touching surface of disk.
- **Poor quality** hardware, including bad processor fan, an overheated hard drive.
- Malware.

Solution:

A **Disk repair utility program** mark **bad sectors** and ensure that file system no longer tries to use them. When files has been affected, utility might be able to recover some of data or it has to delete files from file system.

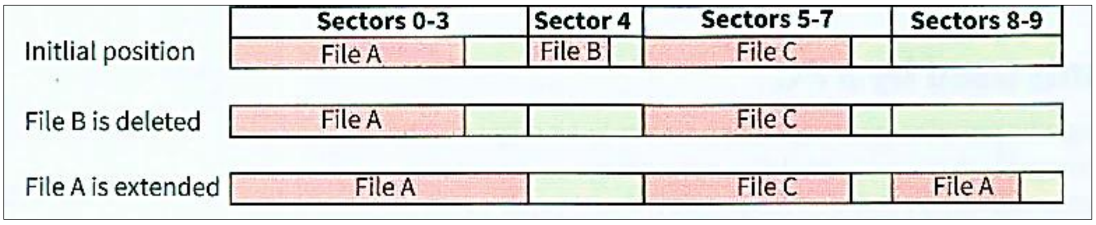
Hard Disk Defragmenter:

Disk defragmenter utility is not primarily concerned with errors. Disk will gradually become **less efficient** because constant creation, editing and deletion of files leaves them in a **Fragmented State**. The cause of this is **logical arrangement** of data in sectors which does not allow a file to be stored as a contiguous entity.

Defragmenter utility program reorganizes file storage to return it to a state where all files are stored in **Contiguous Sectors**.

Working:

Initially **file A** occupies three sectors fully and part of a fourth one. **File B** is small so occupies only part of a sector. **File C** occupies two sectors fully and part of a third. When **File B** is deleted, sector remains unfilled because it would require too much system overhead to rearrange the file organization every time there is a change. When File **A** is extended it completely fills first four sectors and remainder of the extended file is stored in all of Sector **8** and part of Sector **9.** Sector 4 will only be used again if a small file is created or if disk fills up, when it might store the first part of a longer file.



❖ **Backup Software**

Utility program is **Safer** and **Reliable** approach to save **Backup** of data. Utility software control process and store backed-up data into memory stick.
In particular it can do two things:

- Establish a **schedule** for backups
- Only create a new **backup file** when there has been a change.

❖ **File Compression**

- A file compression utility program can be used by an OS to **minimize** hard disk storage requirements.
- File compression is important when transmitting data or to compress a file before attaching it to an email.

❖ **Virus Checker**

- A virus-checking program should be installed to **protect** a computer system.
- When files are being sent from one computer to another it is possible that they may contain a virus which will infect receiving computer.
- Virus checker is a utility program which keeps a constant check on files searching for viruses and deletes it if found.

Program Libraries

Library Programs are collections of **resources** used to **develop software** or program. These include **pre-written code** and **Subroutines**.

Examples: built-in functions

Program libraries are used

- ❖ When software is **under development** and programmer can Utilize **pre-written** subroutines in their own programs, thus saving considerable **development time.**
- ❖ To help software developer who wishes to use **Dynamic Link library (DLL)** subroutines in their own program, so these subroutines must be available at **run time.**

What is a DLL?

In **Dynamic Link libraries (DLL)**, software being developed is not linked to the library routines until actual run time. These library routines would be stand-alone files only being accessed as required by the new program – the routines will be available to several applications at the same time.

Advantage:

By using a DLL, updates are **easier** to apply to each **module** without affecting other parts of program.

Example, you may have a payroll program, and tax rates change each year. When these changes are isolated to a DLL, you can apply an update without needing to build or install whole program again.

Static Libraries:

Software being developed is linked to **executable code** in library at time of compilation. So library routines would be **embedded** directly into new program code.

Pros and Cons of using DLL files	
Pros of using DLL files	**Cons of using DLL files**
Executable code of main program is much smaller since DLL files are only loaded into memory at **run time**.	Executable code is not self-contained, therefore all DLL files need to be available at run time otherwise error messages (such as missing .dll error) will be generated and software may even crash
It is possible to make changes to DLL files independently of main program, consequently if any changes are made to DLL files it will not be necessary to **recompile** main program	Any DLL linking software in main program needs to be available at run time to allow links with DLL files to be made
DLL files can be made available to a number of applications at same time	If any of DLL files have been changed (either intentionally or through corruption) this could lead to main program giving **unexpected** results or even **crashing**.
All of above save **memory** and also save **execution time**.	Malicious changes to DLL files could be due to result of malware, thus presenting a risk to main program following the linking process.

****