HOLY TRINITY GIRLS' SECONDARY SCHOOL

GRADE 10 COMPUTER STUDIES NOTES

OPERATING SYSTEMS-LUKWESA HOWARD

The operating system (OS) plays a crucial role in managing communication between the computer and peripheral devices (like keyboards, printers, mice, hard drives) while also ensuring data integrity. Here's a breakdown of how it does both:

1. **Communication with Peripheral Devices**

The OS communicates with peripheral devices through:

a. Device Drivers

- A **device driver** is software that acts as a translator between the OS and the hardware.
- Each peripheral device has its own driver that allows the OS to send commands and receive data in a format the device understands.

b. Input/output (I/O) Management

- The OS manages I/O operations using **I/O controllers** and buffers to handle data transfers efficiently.
- It schedules data flow to and from devices using techniques like **interrupts** and **polling**:
    - **Interrupts** notify the CPU when a device needs attention (e.g., printer is ready).
    - **Polling** is when the OS checks the status of a device at regular intervals.

c. Plug and Play (PnP)

- Modern OSs use PnP to detect and configure newly connected devices automatically without user intervention.

---

2. Maintaining Data Integrity

The OS ensures data integrity using:

a. File System Management

- It organizes and stores data using file systems (e.g., NTFS, FAT32, ext4), ensuring that data is stored in an orderly and retrievable way.
- It prevents unauthorized access and corruption using permissions and access controls.

b. Error Detection and Correction

- The OS uses techniques like **checksums**, **parity checks**, and **file system journaling** to detect and recover from data errors.

c. Data Caching and Buffering

- Temporary storage (cache or buffers) is used to manage data being read or written, reducing the risk of data loss during sudden shutdowns or device errors.

d. Backup and Recovery Tools

- Built-in utilities or scheduled backups ensure that data can be restored in case of corruption or hardware failure.

Summary

- **Device drivers** and **I/O systems** allow the OS to communicate efficiently with peripherals.
- **File systems**, **error detection**, **access controls**, and **backup mechanisms** help the OS maintain **data integrity**.

Type	File Organization	Access Method
Sequential	Linear order	Read from start to end
Direct (Random)	Hash-based or calculated location	Jump to specific record
Indexed	Uses an index to store pointers	Use index to access efficiently

File Management in Operating Systems

File management is a core function of an operating system (OS) that deals with the creation, organization, storage, retrieval, naming, sharing, and protection of files on a computer system.

◆ What is a File?

A **file** is a collection of related data or information stored on secondary storage (like a hard disk or SSD), which is treated as a single unit by the OS.

◆ Objectives of File Management

The file management system aims to:

- Organize and store files efficiently.
- Allow users and programs to create, read, write, and delete files.
- Provide access control and file protection.
- Enable file sharing among users or processes.

◆ Functions of File Management

1. 📄 **File Creation and Deletion**
    - Enables users to create new files and delete unnecessary ones.
2. 📂 **Directory Creation and Management**
    - Organizes files into directories (folders) for easier access and grouping.
3. ✏️ **File Manipulation**
    - Supports operations like reading, writing, editing, renaming, and copying files.
4. 🔐 **Access Control and Protection**
    - Prevents unauthorized access to files through permissions (e.g., read/write/execute).

5. 🔄 **File Backup and Recovery**
    - Provides mechanisms for backing up data and recovering files in case of failure.
6. 🔗 **File Sharing**
    - Allows files to be shared among multiple users or processes, with controlled access.
7. 🗂️ **File Organization and Storage Management**
    - Determines how files are stored (e.g., sequential, indexed) and manages physical space on storage media.

◆ File Attributes

Each file usually has metadata or attributes managed by the OS, such as:

- Name
- Type (e.g., .txt, .exe)
- Size
- Location on disk
- Creation/modification date
- Access permissions
- Owner/user ID

◆ File Naming Conventions

Operating systems allow files to be identified by names. Naming rules vary by system, but generally:

- May include extensions (e.g., report.pdf)
- Should avoid special characters (like /, ?, *)
- Are case-sensitive or insensitive depending on the OS

◆ Directory Structures

Files are organized in directories to manage large numbers of files. Common directory structures include:

1. **Single-Level Directory** – All files in one directory.
2. **Two-Level Directory** – Separate directory for each user.
3. **Tree-Structured Directory** – Hierarchical, allows subdirectories.
4. **Acyclic Graph Directory** – Allows file sharing through links.
5. **General Graph Directory** – Allows cycles, needs garbage collection.

◆ File Access Control Mechanisms

Operating systems use permission settings such as:

- **Read (r)** – View file contents.
- **Write (w)** – Modify file contents.
- **Execute (x)** – Run the file if it's a program.
  These can be assigned to:
- Owner
- Group
- Others (public)

◆ File Management System Architecture

A typical file management system includes:

- **Logical File System**: Manages metadata and user access.
- **File-Organization Module**: Manages file structure and internal format.
- **Basic File System**: Sends commands to the physical storage.
- **I/O Control**: Deals with hardware device interaction.

🔚 Conclusion

File management is essential for effective data handling in any operating system. It ensures files are easily accessible, secure, and efficiently stored, thereby supporting user and system operations reliably.

Feature	Explanation
Folder (Directory)	Container for organizing files and subfolders
Hierarchical Structure	Tree-like arrangement starting from a root
Management	OS handles creation, deletion, navigation, permissions
Navigation Tools	File Explorer, Terminal/Command Line
Access Control	Managed through permissions and user/group ownership

How the Operating System Communicates with Peripheral Devices and Maintains Data Integrity

---

✅ 1. Communication with Peripheral Devices

Peripheral devices include hardware components like printers, keyboards, mice, USB drives, scanners, and external storage. The operating system acts as an intermediary between these devices and the user/application.

◆ Key Components Involved

1. **Device Drivers**
    - A **device driver** is specialized software that tells the OS how to communicate with a specific hardware device.
    - Each type of device has its own driver (e.g., printer driver, USB driver).
    - The OS loads these drivers to enable proper communication with peripherals.
2. **System Calls & APIs**
    - Applications request hardware access through **system calls** or **application programming interfaces (APIs)**.

- The OS translates these requests into low-level instructions for the device.

3. **Interrupts and I/O Ports**
    - When a device needs attention (e.g., a key is pressed), it sends an **interrupt** to the CPU.
    - The OS uses **Interrupt Service Routines (ISRs)** to handle these signals.
    - For example, pressing a key sends an interrupt, which the OS processes to display the character.
4. **I/O Controllers**
    - Hardware components like **I/O controllers** manage communication between the CPU and peripherals.
    - The OS interacts with these controllers via memory-mapped or port-mapped I/O.

🔄 Data Flow Example (USB Flash Drive)

1. USB is connected → OS detects device.
2. USB driver is loaded.
3. File system on the drive is mounted.
4. User accesses the drive → OS uses driver to read/write data.
5. Safely removing the drive ensures proper data write-back.

✅ 2. Maintaining Data Integrity

Data integrity means **ensuring that data is accurate, consistent, and not corrupted** during processing, storage, or transfer.

◆ How OS Maintains Data Integrity

1. **File System Checks**
    - Uses structured file systems (e.g., NTFS, ext4) that organize and track data reliably.

- On startup or after crashes, tools like **CHKDSK** (Windows) or **fsck** (Linux) check for corruption.

2. **Error Detection and Correction**
    - Uses **checksums** or **parity bits** to detect and sometimes correct data errors.
    - ECC (Error-Correcting Code) RAM is used in critical systems for automatic error correction.
3. **Data Buffering and Caching**
    - OS uses **buffers** to temporarily hold data before writing to disk.
    - Prevents loss during high-speed data transfer.
4. **Access Control and Permissions**
    - Prevents unauthorized changes to files using **read/write/execute** permissions.
    - Helps avoid accidental or malicious data corruption.
5. **Transaction Management (Journaling)**
    - Modern file systems (like ext4 or NTFS) use **journaling**, which logs changes before applying them.
    - This ensures that if the system crashes mid-operation, the file system can recover.
6. **Safe Device Removal**
    - The OS requires "safe removal" of devices like USB drives to flush caches and avoid incomplete writes.
7. **Backup and Recovery Tools**
    - OSes often include tools for **automated backups** and **system restore** to protect data over time.

 Summary Table

Function	Description
Device Drivers	Software that allows the OS to talk to hardware

Function	Description
Interrupts	Signals from hardware for attention
File System Management	Organizes data on disk, maintains structure
Journaling	Logs data operations to protect against corruption
Access Control	Limits file access to prevent accidental/malicious changes
Safe Ejection	Ensures all data is written before removing a storage device
Error Detection Tools	Utilities like CHKDSK, fsck, and ECC for checking and correcting data issues