KARNAVATI UNIVERSITY

Unitedworld Institute of Technology

Operating Systems

Course Code: **71203002004**

Kernel I/O Subsystem – Overview

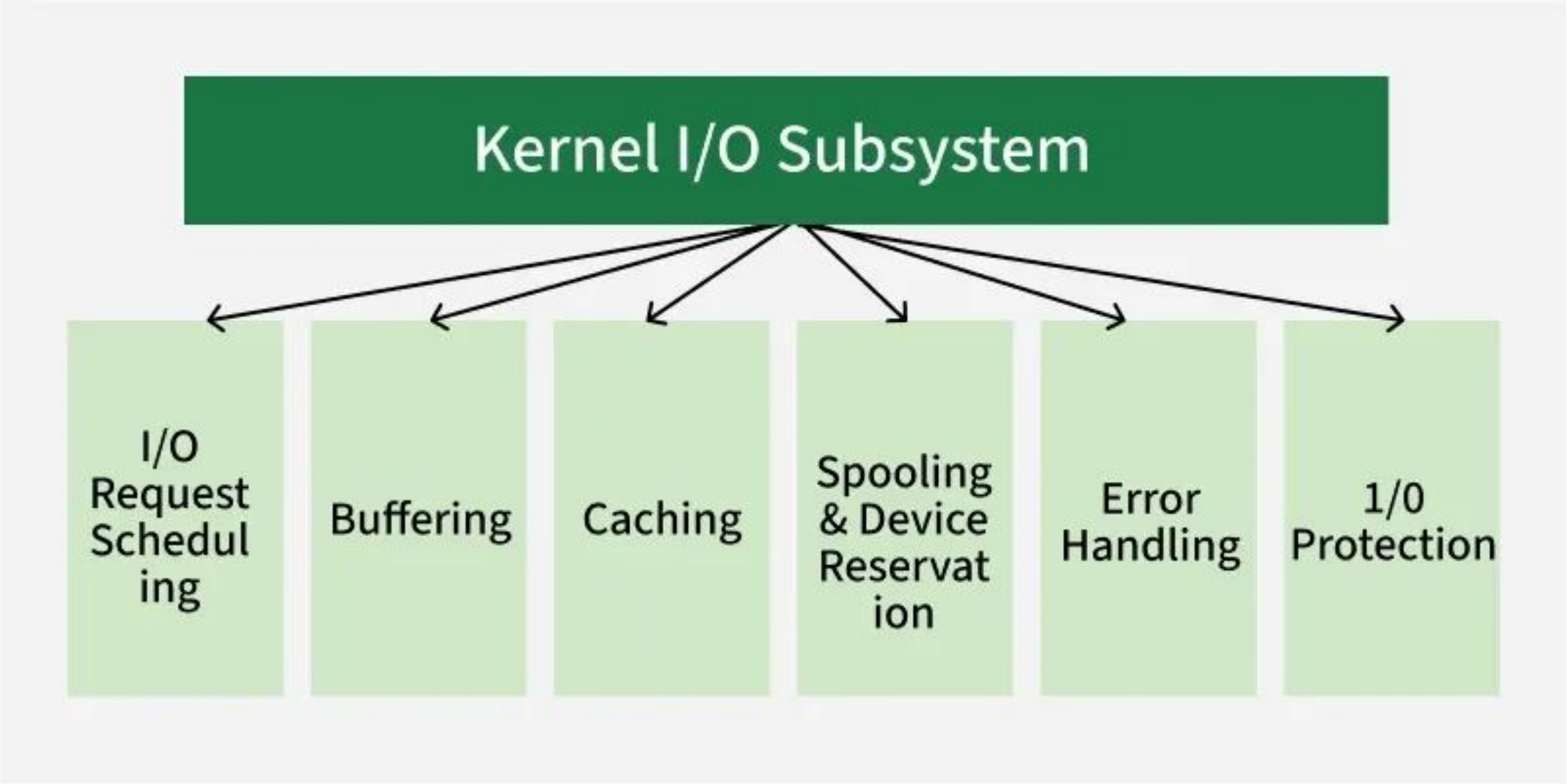
by -

Minal Rajwar

Structure of the Kernel I/O Subsystem

The Kernel I/O Subsystem manages communication between the CPU and all input/output devices (like printers, disks, keyboards, and network interfaces).

It converts high-level I/O requests from user programs into low-level hardware operations.


Kernel I/O Subsystem
I/O Request Scheduling
Buffering
Caching
Spooling & Device Reservation
Error Handling
1/0 Protection

Structure of the Kernel I/O Subsystem

Key Responsibilities:

- **Scheduling:** Organizes I/O requests efficiently.
- **Buffering:** Temporarily stores data during transfer.
- **Caching:** Keeps frequently accessed data for faster retrieval.
- **Spooling:** Manages queued outputs (like print jobs).
- **Error Handling:** Detects and recovers from I/O errors.
- **Protection:** Restricts unauthorized access to I/O devices.

Structure of the Kernel I/O Subsystem

Importance:

- Provides a uniform interface for device access.
- Ensures device independence.
- Handles concurrency among processes.
- Maintains data integrity and system security.

Layers of I/O System Architecture

The I/O system is organized into **layers** to separate high-level operations from low-level hardware control.

Typical Layers:

1. **User-Level I/O**
   - Application programs make I/O requests using system calls (e.g., `read()`, `write()`).
2. **Device-Independent I/O Software**
   - Provides a uniform interface to all devices.
   - Handles buffering, caching, spooling, and naming of devices.
3. **Device Drivers**
   - Translate generic I/O requests into device-specific commands.
   - Control device registers and handle interrupts.

Layers of I/O System Architecture

Typical Layers:

4. Interrupt Handlers

- Manage I/O completion notifications from hardware.
- Wake up processes waiting for I/O.

5. Hardware Layer

- Actual physical devices and controllers perform data transfer.

Goal: Each layer hides the complexity of the one below it, making the system modular and easier to manage.

I/O Request Handling and Buffering Techniques

I/O Request Handling:

1. Application issues an I/O request.
2. Kernel places the request in the device queue.
3. I/O scheduler reorders requests for efficiency.
4. Device driver executes the request.
5. Interrupt handler signals completion to the process.

I/O Request Handling and Buffering Techniques

Buffering Techniques:

- **Single Buffering:** One buffer between device and process.
- **Double Buffering:** Two buffers used alternately to reduce waiting.
- **Circular Buffering:** Multiple buffers arranged in a loop for continuous data flow (useful in streaming).

Purpose of Buffering:

- Manage speed differences between CPU and devices.
- Support different data transfer sizes.
- Maintain data consistency during I/O.

Synchronous vs Asynchronous I/O

Type	Description	Process Behavior	Example
Synchronous I/O	Process waits until I/O completes.	Blocked during I/O operation.	Reading from a keyboard input.
Asynchronous I/O	Process continues execution while I/O happens in background.	Not blocked; notified on completion.	Sending data over a network.

Key Difference:
Synchronous = Waits for I/O completion
Asynchronous = Proceeds without waiting

DISCUSSION & REVISION

1. Which component of the OS manages communication between CPU and I/O devices?
2. What temporarily stores data during transfer between devices?
3. Which layer translates generic I/O requests into device-specific commands?
4. In which type of I/O does the process wait until the operation completes?
5. What stores frequently accessed data to speed up future access?

REFERENCES

1. https://www.geeksforgeeks.org/operating-systems/kernel-i-o-subsystem-in-operating-system/
2. https://www.tutorialspoint.com/kernel-i-o-subsystem-in-operating-system