KARNAVATI UNIVERSITY



Operating Systems

Course Code: **71203002004**

# *Kernel I/O Subsystem – Services & Implementation*

***by -***

***Minal Rajwar***

# Caching, Spooling, and Device Reservation

**Spooling (Simultaneous Peripheral Operation On-Line)**

- **Meaning:** Temporarily stores I/O data (like print jobs) in secondary storage before sending it to slow devices.
- **Purpose:** Keeps the CPU busy by not waiting for slow I/O devices.
- **Example:** Multiple documents sent to a printer are stored in a spool queue and printed one by one (FIFO).

# Caching, Spooling, and Device Reservation

**Why Spooling is Needed**

- **Avoid CPU idle time** – CPU keeps processing other tasks while I/O happens in background.
- **Synchronize I/O** – Manages multiple print jobs without mixing output.
- **Manage speed mismatch** – Handles slow I/O devices compared to fast CPUs.
- **Enable parallelism** – Allows CPU and I/O to work simultaneously.
- **Provide reliable output** – Maintains job order and consistency.

# Caching, Spooling, and Device Reservation

**How Spooling Works**

1. Data sent by CPU → stored in spool area (on disk).
2. When device is ready → data fetched → processed.
3. Uses FIFO queue for fairness and order.
4. Combines buffering (temporary storage) and queuing (job order).

# Spooling vs Buffering

| Feature | Spooling | Buffering |
|---|---|---|
| Works on | Multiple jobs | Same job |
| Storage | Secondary (disk) | Main memory |
| Efficiency | Higher | Lower |
| Remote processing | Supported | Not supported |
| Error handling | Easier recovery (data on disk) | Buffer overflow possible |
| Complexity | More complex | Simpler |

# Spooling

**Pros**

- Improves CPU utilization and efficiency
- Manages multiple user jobs
- Ensures ordered and error-free output

**Cons**

- Needs large storage
- Increases disk traffic
- Delays if spool is full

# Error Handling and I/O Protection

**Error Handling**

1. **Detecting errors:** OS checks for issues (device failure, missing file).
2. **Reporting errors:** Notifies system or app via error messages or exceptions.
3. **Responding to errors:** Retries operation, uses fallback data, or safely terminates faulty process.
4. **Security:** Avoids revealing sensitive info in error messages.

# Error Handling and I/O Protection

**I/O Protection**

- **Privileged Instructions:** User programs **cannot directly access hardware** — only OS can.
- **Kernel Mediation:** Kernel handles all I/O system calls.
- **Goal:** Prevent user programs from corrupting data or crashing devices.
- **Benefits:** Security, stability, and controlled access to hardware.

# Performance Optimization in Kernel I/O

Optimizing how data moves between CPU and storage.

**1. I/O Schedulers**

- **Purpose:** Decide the order of disk I/O requests for best performance.
- **Common Schedulers:**
  - **Noop:** FIFO queue, best for SSDs.
  - **Deadline:** Prioritizes reads with time limits (databases).
  - **CFQ:** Fair I/O sharing for all processes.
  - **mq-deadline:** Scalable version for modern NVMe SSDs.
- **Tuning Parameters:** Adjust values like `read_expire` or `fifo_batch`.

**2. Asynchronous I/O (AIO) and io_uring**

- **AIO:** App continues while I/O runs in background.
- **io_uring:** Modern Linux interface — faster, less kernel overhead.

# Performance Optimization in Kernel I/O

### 3. Caching and Buffer Management

- **Page cache** & **buffer cache** store frequently accessed data in RAM.
- **Tunable Parameters:**
  - `vm.dirty_ratio` – how much RAM to use before writing to disk
  - `vm.swappiness` – how aggressively to use swap

### 4. Disk and Filesystem Optimization

- **Choose right filesystem:** `ext4` (general), `XFS` (large files).
- **Use mount options:** `noatime` (skip access-time updates).
- **Proper disk alignment:** Reduces I/O delays.

# Performance Optimization in Kernel I/O

**5. Hardware Optimization**

- Use **NVMe SSDs** for speed.
- Use **RAID** for redundancy and performance.

**6. Monitoring Tools**

- **iostat**, **iotop**, **atop** – identify I/O bottlenecks and tune accordingly.

# I/O Scheduling Overview within Kernel

**What is I/O Scheduling?**

- Manages **order of I/O requests** to improve performance and fairness.
- Needed because **I/O devices are slower** than CPU.

**Why Needed**

- Reduce wait time
- Prevent long delays
- Optimize disk arm movement
- Fairly share device access among processes

# I/O Scheduling Overview within Kernel

**How It Works**

1. **Request Initiation** – Process sends I/O request.
2. **I/O Traffic Controller** – Tracks all devices and requests.
3. **I/O Scheduler** – Decides which request runs next.
4. **Device Handler** – Transfers data.
5. **Completion** – Notifies program.

# DISCUSSION & REVISION

1. Spooling temporarily stores data in which type of memory before sending it to a device?
2. Which I/O scheduler is best suited for SSDs or virtualized environments?
3. User programs must make system calls for I/O because direct hardware access requires which type of instructions?
4. Which I/O scheduling algorithm serves requests strictly in arrival order?
5. The kernel uses which cache to store frequently accessed disk data in memory?

# 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