KARNAVATI UNIVERSITY

Operating Systems

Course Code: **71203002004**

Process synchronization & Mutual Exclusion

by -
Minal Rajwar

Introduction to Process Synchronization

- Process Synchronization makes sure that multiple processes or threads in a computer can run at the same time without disturbing each other.
- Its main goal is to allow processes to share resources safely, without causing mistakes or data inconsistency. To do this, techniques like **semaphores**, **monitors**, and **critical sections** are used.

In systems with many processes, synchronization helps:

- Keep data correct and consistent
- Avoid problems like **deadlocks**
- Ensure the system works efficiently

Processes can be of two types:

1. **Independent Process** – Runs without affecting other processes.
2. **Cooperative Process** – Can affect or be affected by other processes.

Synchronization problems mainly occur with **cooperative processes** because they share resources.

Process Synchronization

- Process Synchronization means managing how multiple processes run together in a system so they can share resources safely and predictably.
- It prevents problems like **race conditions** and ensures smooth working in concurrent systems

Problems due to improper synchronization.

- **Inconsistency** – Happens when two or more processes change the same data at the same time without coordination, leading to wrong or unreliable results.
- **Loss of Data** – If processes overwrite each other's work without waiting, important data can be lost or corrupted.
- **Deadlock** – When processes are stuck waiting for each other's resources and none can continue, the system stops responding.

Types of process synchronization

- **Competitive** – Processes compete for the same resource. Without proper synchronization, this can cause inconsistency or data loss.
- **Cooperative** – Processes affect each other's work. Without proper synchronization, this can cause deadlocks.

Example:

ps | grep "chrome" | wc

ps > List running processes

grep > Find chrome from the ps output

wc > count word from the grep output

Here, processes **work together** (ps → grep → wc). This is **cooperative synchronization**, and the OS manages such interactions.

When is Synchronization Needed?

- **Critical Section** – Part of the program that accesses shared resources. Only one process should run it at a time.
- **Race Condition** – Output depends on the order processes finish updates. Synchronization ensures the correct order.
- **Preemption** – If a process using a shared resource is paused for another process, the second process might get wrong data without proper synchronization.

Race Condition

- A race condition happens when the output of a program depends on the order in which multiple processes or threads run in a **critical section** (a part of code that uses shared resources).
- If processes access shared data at the same time without proper control, they can overwrite each other's changes, causing **incorrect results**.

How to avoid it:

- Make the critical section **atomic** (runs completely without interruption)
- Use **locks** or **atomic variables** for synchronization

Example:

- Shared variable: balance = 100
- deposit(10) → balance = balance + 10
- withdraw(10) → balance = balance - 10

Interleaving problem:

1. deposit() reads 100 → gets paused
2. withdraw() runs → balance becomes 90
3. deposit() continues → sets balance to 110

Final balance becomes **110 instead of 100** — this is a race condition.

Mutual Exclusion in Synchronization

- When multiple processes run at the same time, they sometimes need to use **shared resources** (like files, printers, or shared data). The part of the program that accesses these resources is called the **critical section**.
- If two processes enter the critical section at the same time, they can cause **inconsistent data** or **race conditions**.

Mutual Exclusion means:

- Only **one process** can be inside the critical section at any given time.
- This idea was introduced by **Dijkstra** and is a key requirement to avoid race conditions.

When is Mutual Exclusion Needed?

Whenever a shared resource is being accessed — for example:

- Files
- Printers (I/O devices)
- Shared data structures

If Process P1 is using Resource R1, Process P2 must wait until P1 is done.

Conditions for Mutual Exclusion

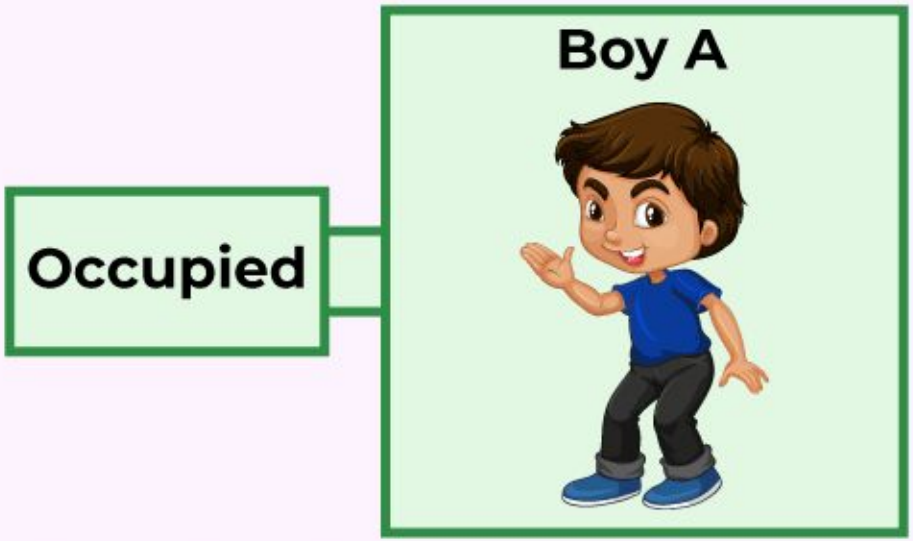
- Only one process can be in its critical section at a time.
- Don’t depend on process speed (fast or slow).
- A process outside the critical section should not block another from entering.
- Every process must get a turn in **finite time** (no endless waiting).

Ways to Implement Mutual Exclusion

- **Software methods** – Processes control it themselves (slow, error-prone).
- **Hardware methods** – Special CPU instructions (fast but incomplete, may cause deadlocks).
- **Programming language support** – Provided by OS or language features (e.g., locks, synchronized blocks).

Example : Supermarket Changing room.

- Boy A enters the changing room → “Occupied” sign is ON.



Since Boy A is Inside the changing room, the sign on it prevent the other from entering the room.

Boy B has to wait outside the changing room till boy A comes out

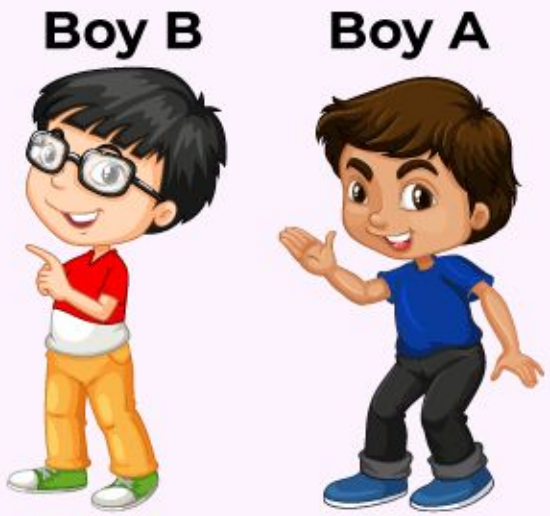
Example : Supermarket Changing room.

- Boy B waits until Boy A finishes.
- When Boy A leaves → “Vacant” sign is ON, Boy B enters.

VACANT

Changing room initially lies vacant, allowing people to enter the room and use it

Since Boy A is done using the changing room, he vacates it- making the sign outside the room change.



Since the changing room has been vacted by Boy A, Boy B can enter the hanging room.

Example : Supermarket Changing room.

Here:

- Changing room = Critical Section
- Boy A and Boy B = Processes
- Sign = Synchronization mechanism

DISCUSSION & REVISION

1. Who introduced the concept of mutual exclusion?
2. What is the part of the program that accesses shared resources called?
3. Which device is an example of a shared resource?
4. What problem does mutual exclusion prevent?
5. Which method of mutual exclusion uses CPU instructions?

REFERENCES

1. https://www.geeksforgeeks.org/operating-systems/introduction-of-process-synchronization/
2. https://www.geeksforgeeks.org/operating-systems/mutual-exclusion-in-synchronization/