

## Assignment 03

Q1 Explain it.

Ans 1 :- A race condition occurs when multiple people try to access or change a shared resource at the same time, causing inconsistent results.

Q2 compare dependency.

Ans 2 :- Peterson's solution is software-based and simple but works only for two processes and relies on strict hardware ordering. Semaphores are more flexible, support multiple processes, and depend on hardware atomic instructions, like test-and-set.

Q3 The producer system.

Ans 3 :- Monitors automatically handle synchronization using built-in locking, reducing programmer errors. They also ensure that only one thread executes monitor code at a time, improving safety across multiple CPU cores.

Q4 For the it.

Ans 4 :- starvation occurs when continuous readers prevent writers from ever accessing the shared resource.

Using fair scheduling or queue-based ordering ensures every reader/writer gets a turn.

Q5 In deadlock OS.  
Ans :- Processes must request all required resources at once, leading to low resource utilization. This can cause unnecessary delays because resources stay reserved even when not immediately needed.

Q7 Dining philosophers - to avoid it.

\* Deadlock - Prone Scenario (The problem)  
Deadlock occurs when all 5 philosophers ( $P_0$  to  $P_4$ ) simultaneously follow the standard procedure:  
Action  $\rightarrow$  each philosopher acquires their left chopstick (semaphore  $S_i$ ) and then waits for their right chopstick ( $S_{i+1} \pmod 5$ ).

Result  $\rightarrow$  A circular wait occurs.  $P_0$  holds  $S_0$  and waits for  $S_1$  (held by  $P_1$ ),  $P_1$  holds  $S_1$  and waits for  $S_2$  (held by  $P_2$ ), ..., and  $P_4$  holds  $S_4$  and waits for  $S_0$  (held by  $P_0$ ).

Outcome  $\rightarrow$  All philosophers are blocked indefinitely.

\* Deadlock - Free Semaphore Solution  
To prevent the deadlock, we must break the circular wait condition. A simple, effective solution is to limit the number of philosophers allowed to attempt eating concurrently.  
Mechanism  $\rightarrow$  Introduce a counting semaphore, max-eating, initialized to 4 (one less than the number of philosophers).

- Logic  $\rightarrow$  A philosopher must successfully acquire max eating before attempting to pick up the first chopstick.
- Result  $\rightarrow$  At most 4 philosophers can proceed to pick up chopsticks.

counting semaphore  $\text{max-eating} = 4;$   
do {

```
wait (max-eating); // Acquire permission
wait (S[i]); // Pick up left chopstick (S[i])
wait (S[(i+1) mod 5]); // Pick up Right chopstick
signal (S[(i+1) mod 5]); // Put down right chopstick
signal (S[i]); // Put down left chopstick
signal (max-eating); // release permission
```

- Outcome :- If 4 philosophers hold one chopstick each, 5<sup>th</sup> chopstick remains free. When one of the 4 finishes eating and releases the max-eating semaphore, another philosopher is guaranteed to proceed, thus preventing the system from entering the deadlocked state.

Ques 8 An OS uses \_\_\_\_\_ rate.

- Interrupt handling time = 5  $\mu$ s per interrupt
- Device transfer rate = 500 KB/s
- Data per interrupt = 100 bytes.

Date .... / .... / .....

(a) CPU time spent handling interrupts per second

\* Number of interrupts per second :-

$$\text{Interrupts/sec} = \frac{500000 \text{ bytes/sec}}{100 \text{ bytes}} = 5000$$

\* CPU time spent :-

$$5000 \text{ interrupts/sec} \times 5 \mu\text{s} = 25000 \mu\text{s} = 25 \text{ ms}$$

CPU time spent = 25 ms per second (2.5% of CPU time)

(b) Improvement to reduce CPU overhead

Use DMA (Direct Memory Access) so data transfer occurs in large blocks without generating frequent interrupts.

This reduces interrupt rate while keeping the same transfer speed.

Ans

Shubh  
21/11/21