COMPUTER SCIENCE & IT

OPERATING SYSTEMS

Process Synchronization/ Coordination

Lecture No- 03

Dr. KHALEEL KHAN

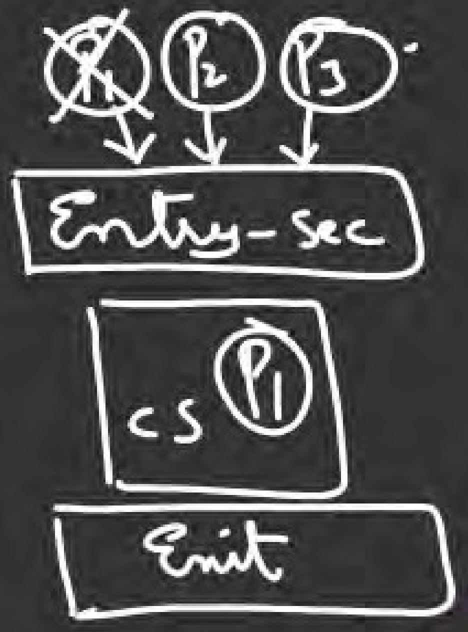
TOPICS COVERED

1. Strict Alternation
2. Peterson Algorithm

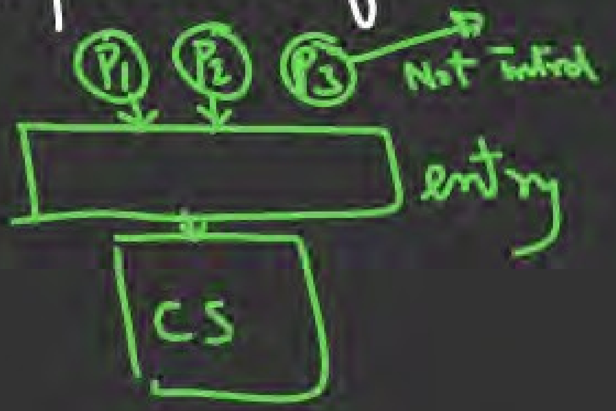
Session-II : Requirements of CS Problem/Synch. Mechanism

Inconsisten
safe
wrz

1) Mutual Exclusion: No Two processes may be Simultaneously presen in their CS;
Primary-Requirement



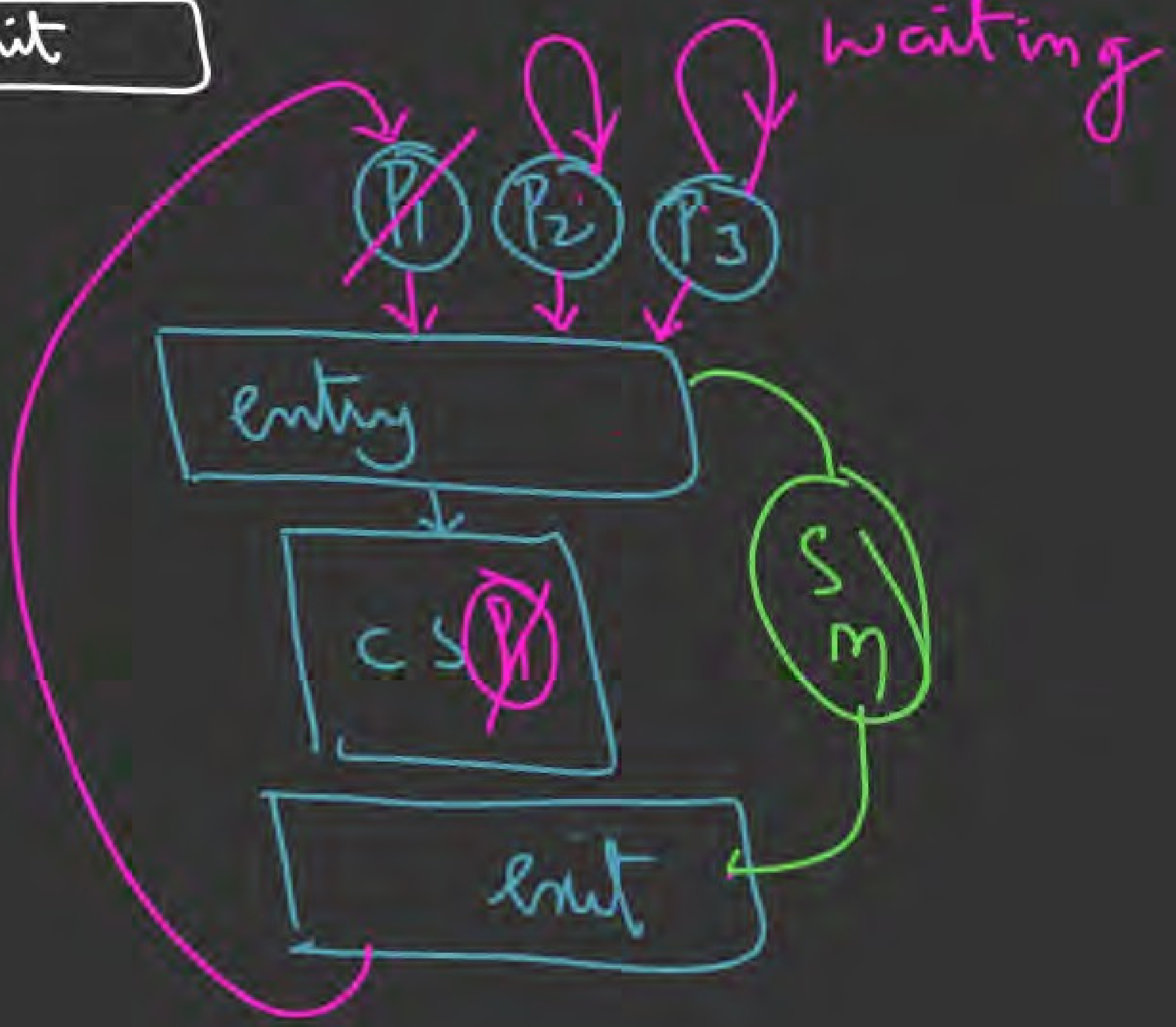
2) Progress: No process running out side (Non cs) the CS, should block/Prevent/Influence the other Interested processes from entering CS;



3) Bounded waiting: No process has to wait for ever to access "CS";

dissatisf.
STARVATION

"There must be a bound on the no. of times that a process is allowed to enter CS, before other process request is satisfied"



Synch. Mechanisms

- **Busy-waiting** (loops)
  - while (count == 0);
- **Blocking / Non-Busy-waiting**
  - if - then - else

S/W ← User-mode
- → Lock-variable
- → Strict-Alternation
- → Peterson soln

Busy waiting

H/W
- → TSL Instn
- → SWAP "

Busy-waiting

OS-Based (Blocking Mechanisms)
- → Sleep-wakeup
- → SEMAPHORE
- → MONITORS*

Assumptions for S.Ms.

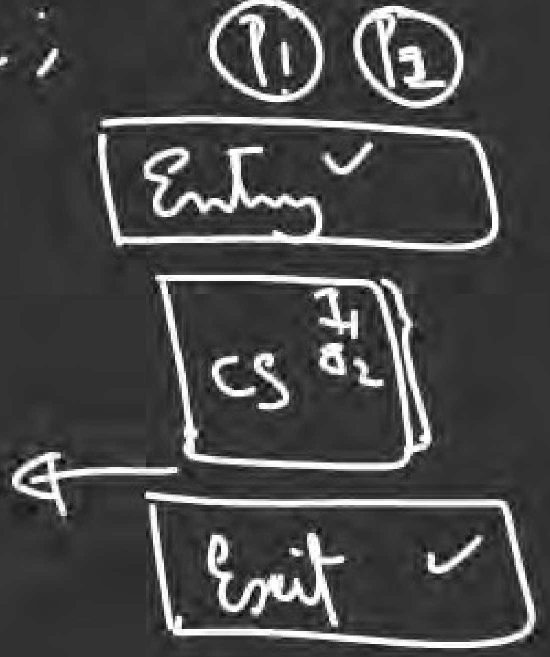
1) PreEmption of Process can happen when it is executing in Entry, Exit or CS;

2) we assume CS is totally error-free;

3) Every Process enters CS & comes out of it, in Finite Time;

4) Process can enter CS only after Completing entry Section

5) Process is said to have left "CS" only if it Completes exit Section;



6) If a process gets PreEmpted from CPU while executing 'CS' Code, then still the process is said to hold "CS";

1) LOCK VARIABLE:

→ User Mode s/w Soln;
→ Busy-waiting; (loop)
→ Multi-process Soln;

Ideology: P_i

Lock → 0 → cs is free
Lock → 1 → cs is in use

cs

Exit: lock = 0

I. H.L.I: ✓

```
int Lock=0;
Void Process(int i)
{
  while(1)
  {
    a) Non_CS();
    b) while(lock!=0);   // Entry
    c) lock=1            // Entry
    d) <cs>
    e) lock=0;           // Exit
  }
}
```

P_1

II. L.L.I: Process: ✓

JNZ: Jump If NOT 0

Entry:

I. Non_cs;
II. Load R_i, Lock;
III. Cmp R_i, #0;
IV. JNZ step II
V. Store lock, #1

VI. <cs>
VII. store lock, #0;

Bounded Waiting

Analysis: ME; Prog; B.W

RQ: ~~P_1~~ P_2 ✓

R_1: 0
R_2: 0

CPU: P_1

lock: 1 ~~0~~ ~~1~~

cs: P_2, P_1

t_1: P_1: I; II; III; IV; Pre

t_2: P_2: I; II; III; IV; V; VI; Pre

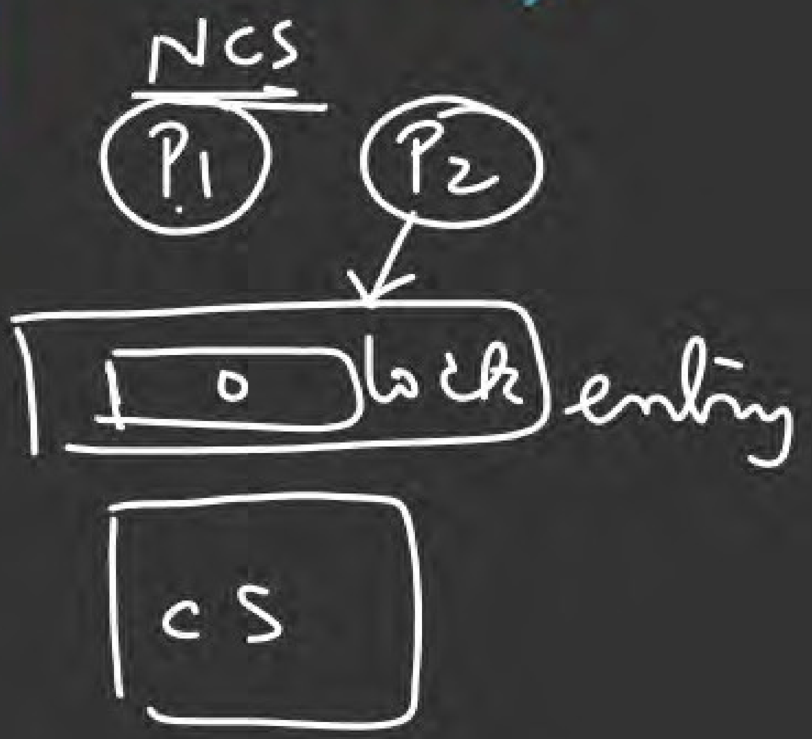
t_3: P_1: V; VI

"Violation of Mutual Exclusion"

Progress

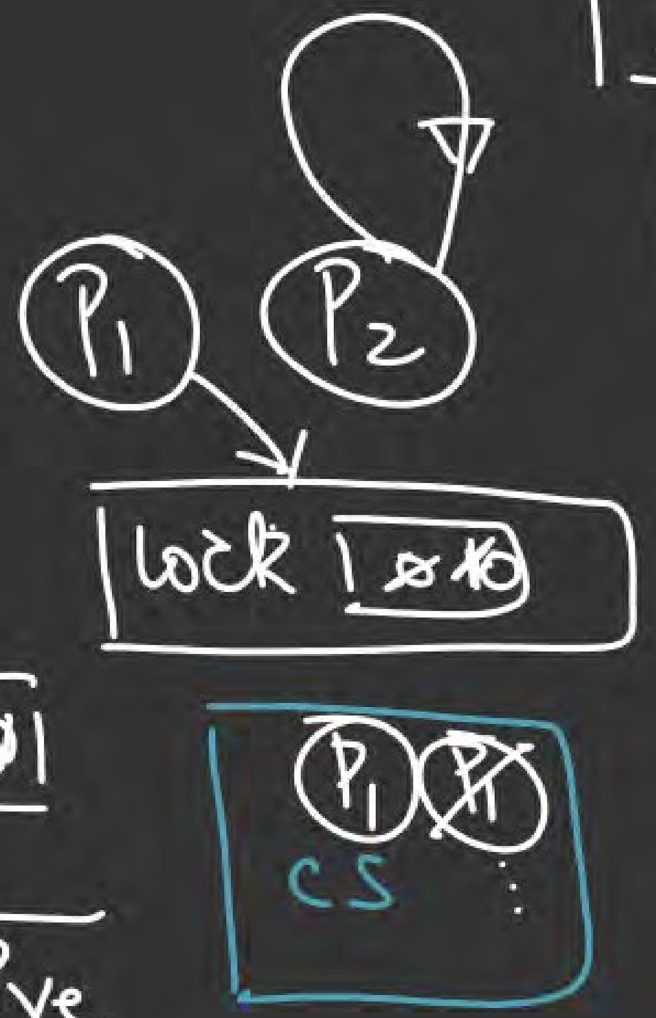
→ Lock variable fails to guarantee Mutual Exclusion.

(Does not guarantee M/E always)



→ Since no process executing in NCS, will block other process from entering CS

∴ Progress is always guaranteed.



→ Does NOT guarantee Bounded wait.

t_1: P_1: I; II; III; IV; V; VI

Pre

t_2: P_2: I; II; III; IV;

Pr

t_3: P_1:

II. STRICT ALTERNATION:

→ Busy-waiting

→ S/W Soln @ User Mode

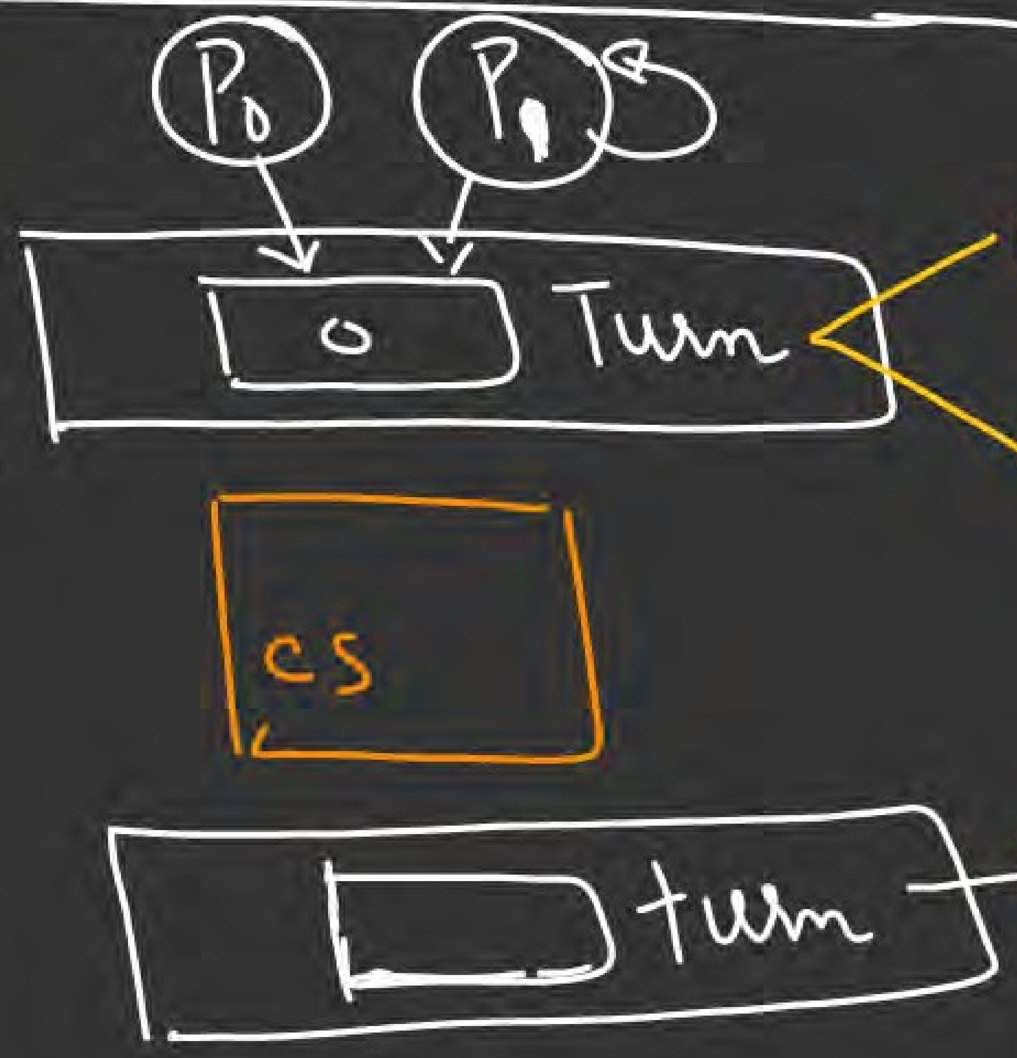
→ 2-process Soln (P_0; P_1)

$\langle P_i \mid P_j \rangle$

→ The value of turn indicates, which process turn to enter CS

Strictly on alternate basis, processes takes turn to enter CS

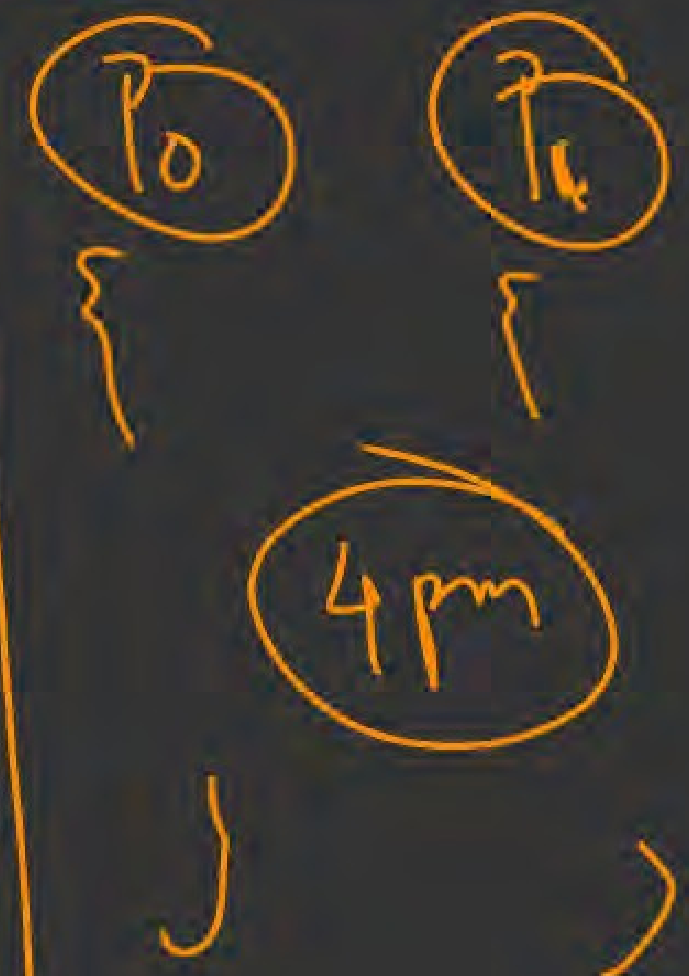
Ideology:



0 → P_0's turn to enter CS

1 → P_1's turn to enter CS

→ Make the value of turn to the id of other Process;



THANK YOU!