CS & IT ENGINEERING

Operating System

Process Synchronization

Lecture No. 1

By- Dr. Khaleel Khan sir

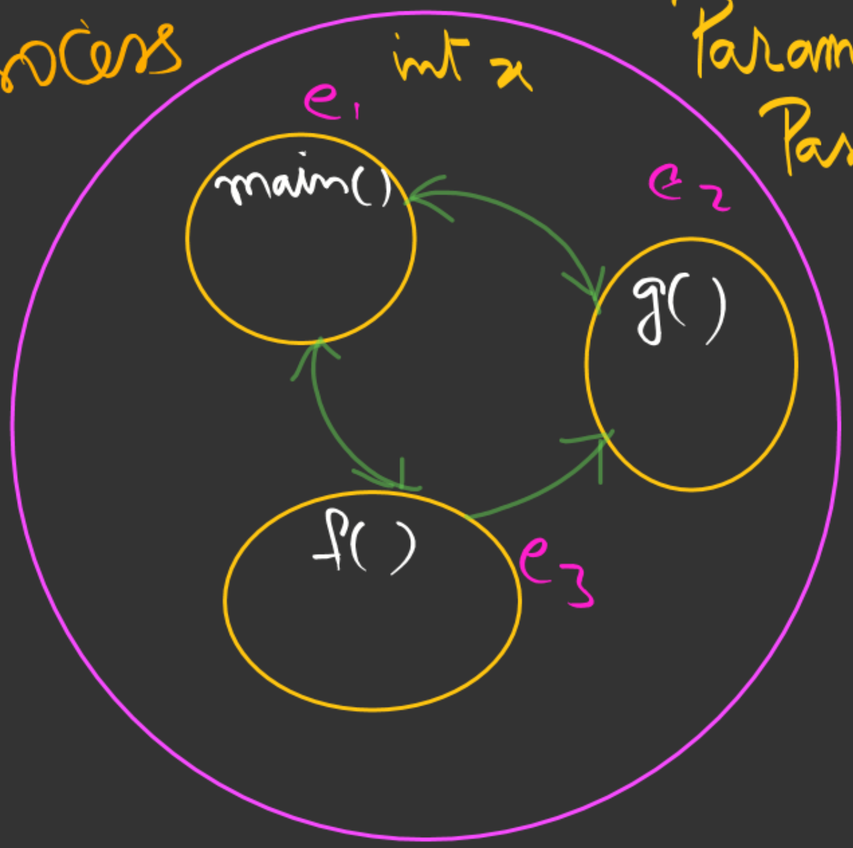
- What is IPC
- What is Synchronization
- Need and Types of Synchronization

What is I.P.C

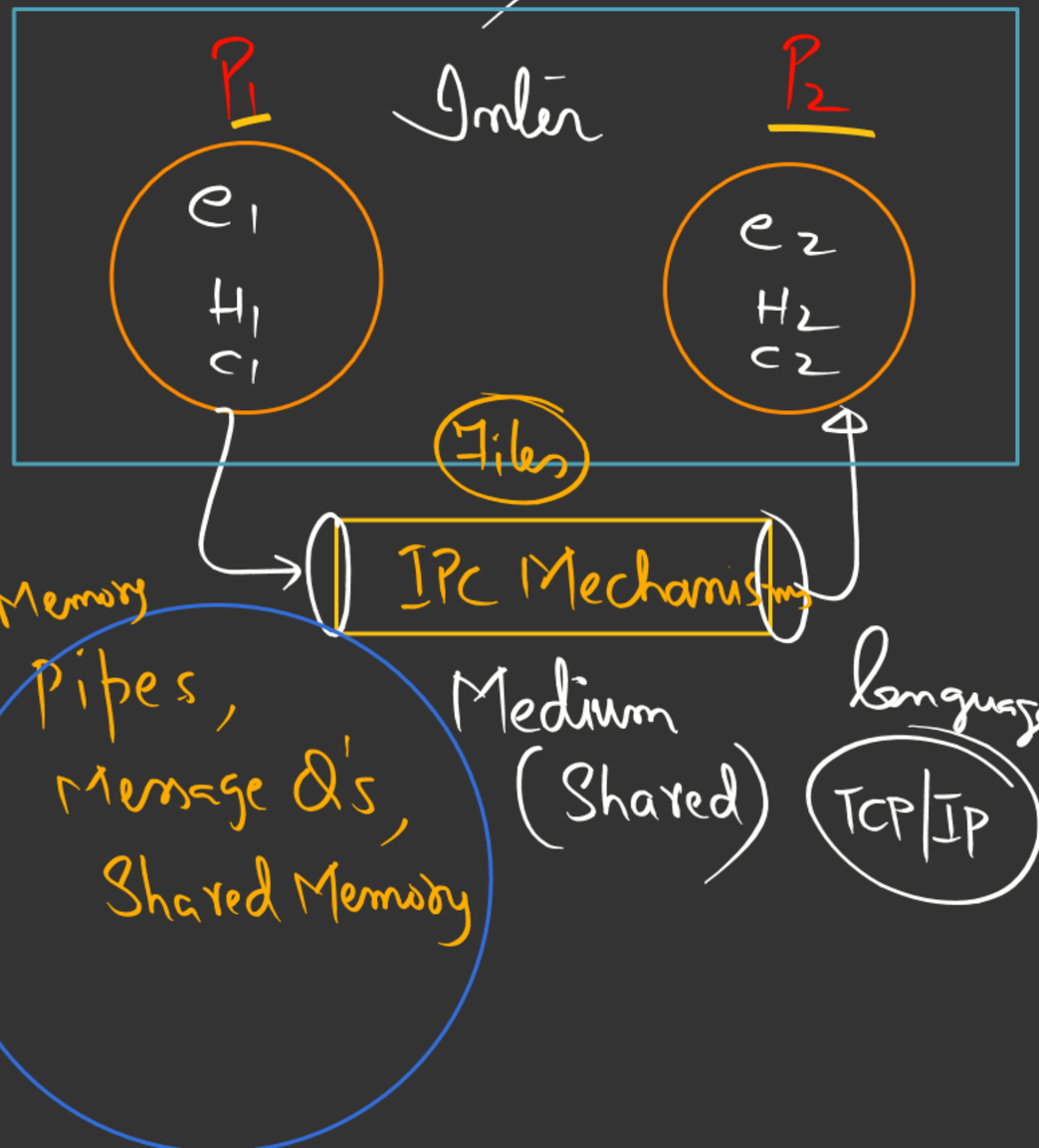
Inter Process Communication

Intra (Global var's)

Process



Parameter Passing



What is Synchronization / Coordination ?

Need for Synchronization

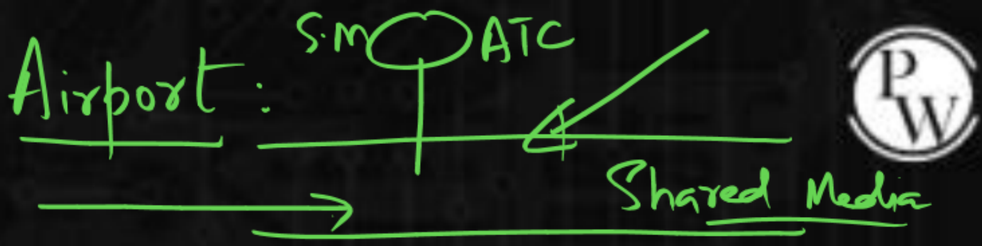
Problems due to lack of Synchroniz. in IPC Environment

(i) Inconsistency (Incorrectness)

(ii) Loss of data

(iii) Deadlocks (undesirable)

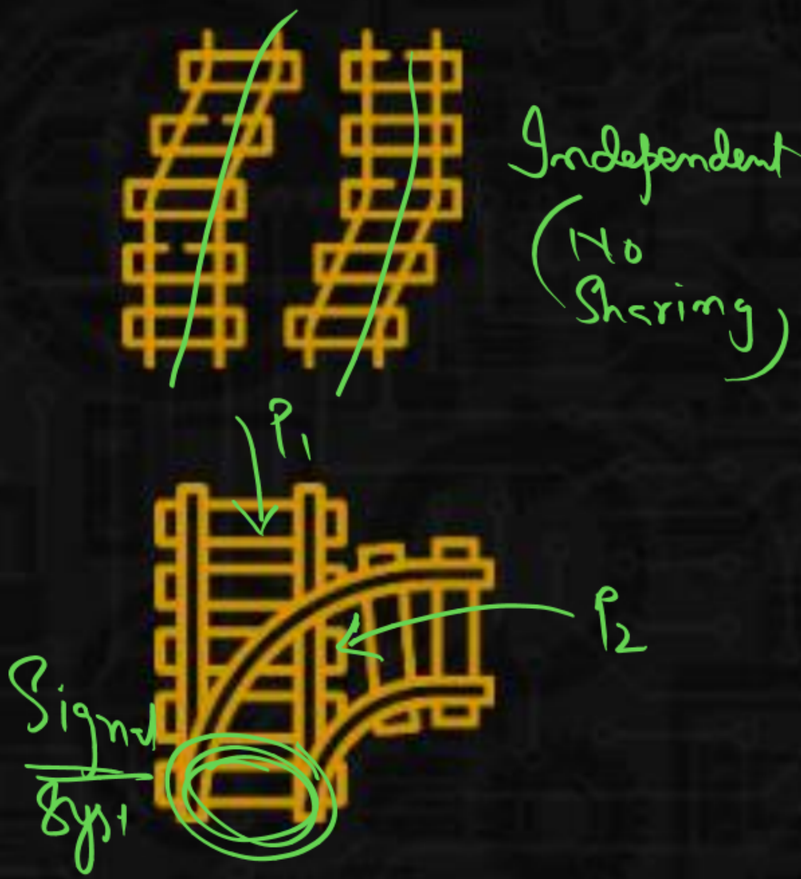
Process Synchronization



Synchronization involves the orderly sharing of system resources by processes.

We can think of this intersection as a system resource that is shared by two processes.

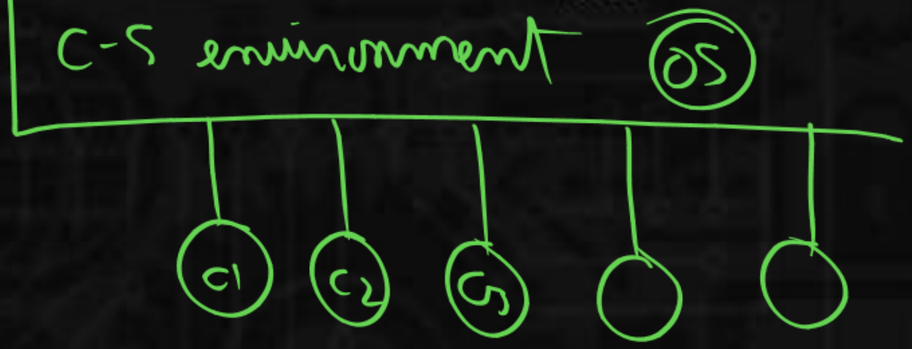
- The car process and the train process
- One process is active at a time - No conflict
- Both process are active - Conflict



Consider a machine with a single printer:

Depending upon whether the printer is already being Used by another process or not, the operating system must decide whether:

- To grant the printing request (if the printer is free),
- Or to deny the request and classify the process as a waiting process until the printer becomes available.

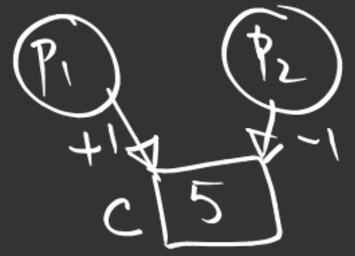


Types of Synchronization [IPC Environment]

(i) Amazon Purchase
(ii) Reservation Systems
(iii) Net Banking

Competition

Two/more processes are in Competition Synchr. iff they Compete/Contend for the accessability of a Shared resource

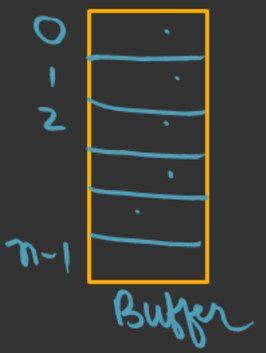


(i) Inconsistency
(ii) Data loss

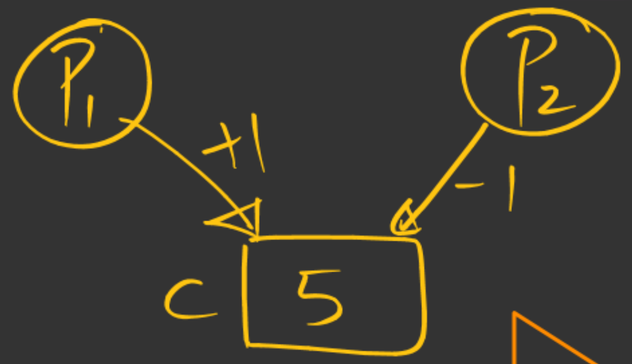
Cooperation → Deadlocks

Two/more Processes are said to be in Cooperative Synchronization, iff they get affected by each other;

Ex: Producer-Consumer Problem



Case Study to demonstrate Inconsistency in IPC Environment




```
int c;
c=5;
```

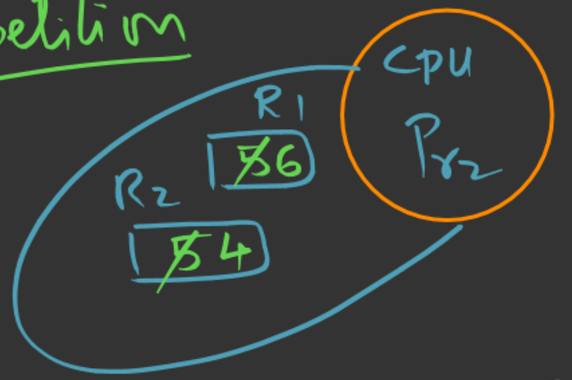

```
{
c=c+1;
}
```

```
{
c=c-1;
}
```

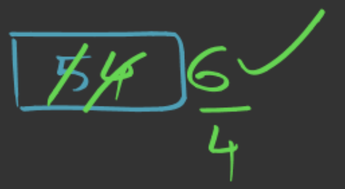
$\bar{I}_1$: Load R_1, c
$\bar{I}_2$: Inc R_1
$\bar{I}_3$: Store c, R_1

J_1: Load R_2, c
J_2: Dcr R_2
J_3: Store c, R_2

Competition



Analysis



t_1: $\underline{Pr_1}$: $\bar{I}_1$; $\bar{I}_2$; Pre

t_2: Pr_2: J_1; J_2; J_3

t_3: Pr_1: $\bar{I}_3$

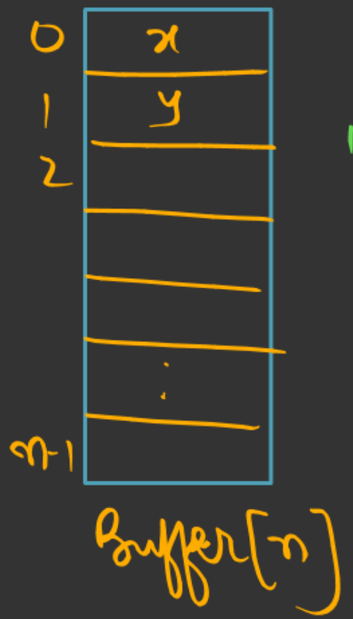
Producer-Consumer : #define n 100

```
int Buffer[n];
int Count = 0;

void Producer(void)
{
    int itemp, in = 0;
    while(1)
    {
NCS a)  itemp = Produce_item();
CS  b)  while (Count == n);
CS  c)  Buffer[in] = itemp;
NCS d)  in = (in+1) % n;
CS  e)  Count = Count + 1;
    }
}
```

in = 2

out = 0



Competition

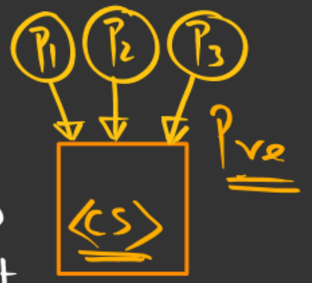
```
void Consumer(void)
{
    int itemc, out = 0;
    while(1)
    {
a)  while (Count == 0);
b)  itemc = Buffer[out];
c)  out = (out+1) % n;
d)  Count = Count - 1;
e)  Process_item(itemc);
    }
}
```

Terminology | Necessary Conditions for Problems of Synchronization in IPC Environ.

(i) **Critical Section (CS)**: is that part of the Program/Process, where Shared resources are accessed;
(Shared resources)

Non-critical Section (remainder): is that part of the Program that does not access Shared resource;

(ii) **Race Condition**:
Situation where Processes are trying to access "CS" and the final result depends on who update last/first

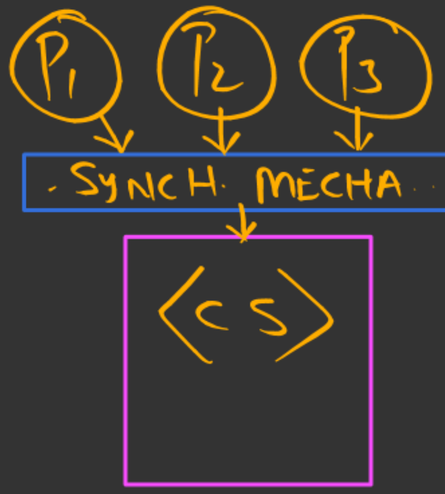


(iii) **PreEmption**:

Critical Section Problem

Soln (Algo)

Synch. Mechanism
S.M



(Model/Arch of CS Problem)


THANK YOU
GW
SOLDIERS !