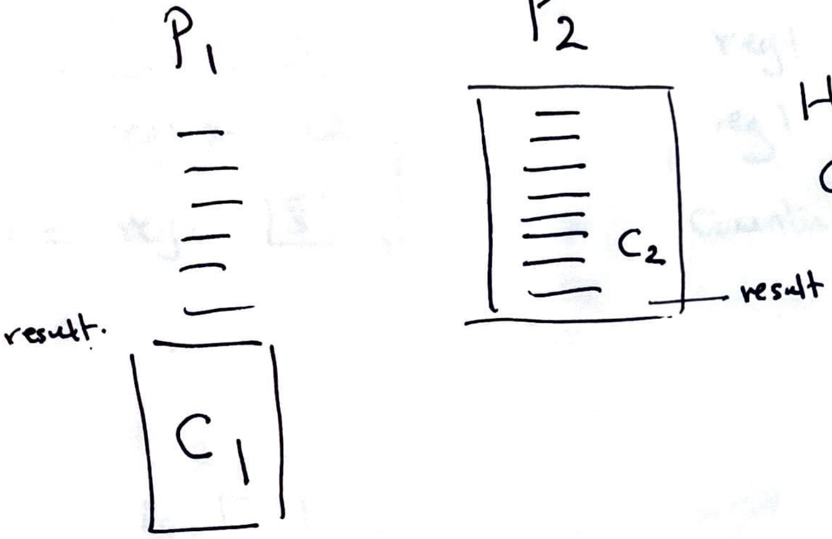
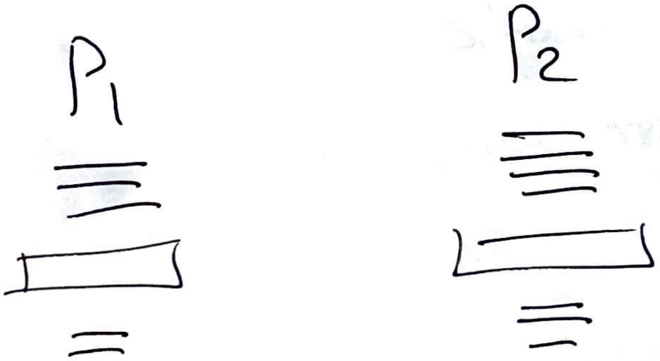
Process Synchronization



How to Sync and Coordinate between processes?

Shared data or shared resources.

global variable =



Counter = 10 // global variable.

P1 P2

Counter++; Counter--;

Counter = 10

P1

Counter ++;

reg1 = counter $\boxed{1}$

reg1 = reg1 + 1 $\boxed{2}$

Counter = reg1 $\boxed{3}$

P2

Counter --;

reg1 = counter $\boxed{4}$

reg1 = reg1 - 1 $\boxed{5}$

Counter = reg1 $\boxed{6}$

$\boxed{4}$	$\boxed{1}$	$\boxed{2}$	$\boxed{3}$	$\boxed{5}$	$\boxed{6}$	⇒ 9
10	10		11	9	9	
$\boxed{1}$	$\boxed{4}$	$\boxed{5}$	$\boxed{6}$	$\boxed{2}$	$\boxed{3}$	⇒ 11

Race Conditions: situation in which multiple processes access and manipulat a shared data concurrently and the outcome depends on the order of execution

Can happen with shared resources.

The Critical Section Problem (CS).

Define the Segment of Code that changes Shared data/resources as the critical Section

Process

```
do {

        entry Section

        | Critical Section |  . . . .        Counter++;

        exit Section .

        | Remainder Section |

} While (TRUE);
```

No 2 processes can execute ther critical Sections at the Same time.

A Solution (to CS) must provide / ~~sta~~ satisfy 3 requirements:

1) **Mutual Exclusion**: Only 1 process at a time can be executing inside their C.S.

2) **Progress**: Only processes wishing to enter the C.S decide who gets entry.

3) **Bounded Waiting**:

No process should starve

bound on the number of times other processes are allowed to enter while a process is waiting

How to Contol access to shared data/resources?

1 Disable Interrupts.

```
P_i
    do{
            disable interrupts( );
            C.S
            Enable-interrupts( );
            R.S
    }while (True);
```

works on a single processor system.

Degrades the efficiency of the system.

2 Simple hardware Instructions

Atomic : Cannot be interrupted while executing this instruction.

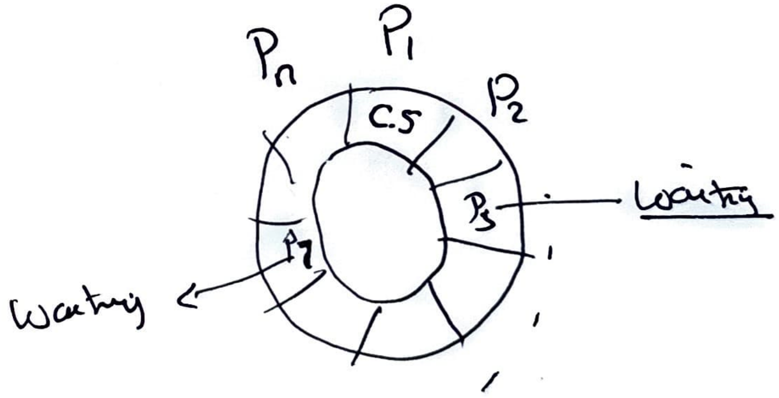
a Test And Set

```
boolean TestAndSet (boolean * target){
    boolean rv=*target;
    * target =TRUE;
    return rv;
}
```

Lock = False;

```
Pi do{
    while(TestAndSet(&lock)); //nothing.     entry

        C.S

        Lock = False;     exit

        R.S

}while(TRUE);
```

IDEA


Pn
P1
C.S
P2
P3
Waiting
P7
Waiting